PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

CURSO DE ENGENHARIA DE COMPUTAÇÃO

Sistema de Gerenciamento Remoto para *Vending Machines*

Projeto Físico, Relatório e Implementação Final – 2° bimestre

Glauco Cattalini Lins
Ricardo Sales da Veiga
Rodrigo Govêa Baran

Professor Orientador:
Afonso Ferreira Miguel

Curitiba
2010

Conteúdo

Resumo....3

Introdução....3

Alterações no Escopo do Projeto....4

Detalhamento Físico e Tecnologias Utilizadas....6

1. Central de Telemetria....8
   - 1.1. Status....9
   - 1.2. Inicializar....11
   - 1.3. Estoque....12
   - 1.4. Ordens de Serviço....12
   - 1.5. Recarga....13
   - 1.6. Alertas....16
   - 1.7. Vendas....17

1. Servidor....18
   - 1.1. Identificação do Cliente....18
   - 1.2. Identificação da Máquina....18
   - 1.3. Estoque de Produtos....19
   - 1.4. Estoque das Máquinas....19
   - 1.5. Relatórios....20
   - 1.6. Atualização de Informações....21
   - 1.7. Troca de Arquivos....21

2. Funcionamento do Servidor....22
   - 2.1. Cadastro de Usuários e Login 22
   - 2.2. Menu de operações básicas....24
   - 2.3. Cadastro de *Vending Machines* 25
   - 2.4. Cadastro de Produtos....26
   - 2.5. Registro de estoque das *Vending Machines*....27
   - 2.6. Emissão de Relatórios....27
   - 2.7. Atualização de Informações....29
   - 2.8. Troca de Arquivos....29

3. Emulador MDB....31
   - 3.1. Funcionamento do Emulador 31
   - 3.2. Usando o Emulador....31

Tecnologias utilizadas....32

1. Transmissão de Dados....32
2. Sobre o Hardware....33
   - 1.1. Conectando as placas....41

2. Sobre o Firmware....44
   - 2.1. Rotina de Inicialização....44
   - 2.2. Navegação no Menu....44
   - 2.3. Conexão com a Internet....45
   - 2.4. Vendas....46
   - 2.5. Alertas e Sensores....46
   - 2.6. Ordens de Serviço....46
   - 2.7. Escrita no LCD....47
   - 2.8. Conversor 9bits(MDB) -> 8bits 47

3. Barramento MDB (Multi-Drop Bus) 48

Procedimentos de Teste e Validação Durante o Desenvolvimento....49

1. Modem GPRS....49
2. Módulo de Telemetria....50

Procedimentos de Teste e Validação Para a Defesa do Projeto....54

1. Procedimentos Executados exclusivamente no Servidor....54

2. Procedimentos executados exclusivamente na *Vending Machine*...55

3. Procedimentos executados em conjunto – VM e Servidor....55

Análise de Riscos....56

Viabilidade e Conclusões....57

Cronograma....59

Referências Bibliográficas....60

Anexos.....61
Anexo 1: Firmware do Modem G24-JAVA 61
Anexo 2: Firmware do Conversor 9bits (MDB) -> 8bits.....97
Anexo 3: Emulador MDB: 101
Anexo 4: Layouts e Esquemáticos das placas:.....104
Anexo 5: Manual de Instruções: 109
Anexo 6: Manual de Integração.....115

Resumo

Vending Machines(VM) são máquinas que permitem disponibilizar produtos ou serviços em regiões de grande movimento sem a necessidade de um operador, no entanto isto exige visitas recorrentes para verificar o estoque dos produtos e/ou funcionamento da máquina.

O Sistema de Gerenciamento Remoto faz uso da tecnologia GPRS para monitorar a situação das máquinas. Tal funcionalidade permite a redução de custos com deslocamento além de um conhecimento imediato da situação de todas as máquinas da empresa via sistema Online, que ficará disponível via Internet.

Para permitir a apresentação do sistema de gerenciamento remoto, um protótipo de VM será apresentado fazendo uso deste dispositivo, permitindo a atualização instantânea das informações cadastradas em um servidor remoto.

Introdução

Através do software do Sistema de Gerenciamento Remoto que foi instalado em um servidor remoto, são coletadas informações suficientemente confiáveis que permitam escolher e definir um catálogo de produtos para cada VM seja ele baseado por região ou demanda histórica da máquina. No discorrer desse documento, a controladora desenvolvida será chamada de módulo de telemetria.

Esta operação permitirá às empresas administradoras de VM's conhecer a demanda de cada máquina e ajustar seus estoques adequadamente, reduzindo visitas técnicas e permitindo o estabelecimento de rotas de reabastecimento mais eficientes, uma vez que se terá o conhecimento prévio sobre a quantidade de itens em cada máquina.

Com base no plano de projeto apresentado em Abril, o escopo do projeto sofreu uma ligeira modificação com relação ao hardware que será desenvolvido. Em função da existência de barramentos e protocolos já documentados, torna-se o objetivo do trabalho apresentar um módulo de telemetria já funcional, que possa ser facilmente instalada em VM's que façam uso do barramento MDB (Multi-Drop Bus). Deste modo o módulo interpreta facilmente o conjunto de comandos necessários para seu funcionamento.

Está contido também no escopo deste projeto o desenvolvimento de um servidor online onde as informações poderão ser consultadas. O servidor é capaz de gerar respostas com formatação XML permitindo a troca de informações, possuindo apenas uma simples interface gráfica. Essa abordagem permite que diferentes desenvolvedores de software integrem em seus sistemas as funcionalidades de telemetria oferecidas.

Não compõem o escopo do projeto as seguintes atividades:

- Desenvolvimento de uma *Vending Machine*

- Desenvolvimento de um modem GPRS
- Interpretação completa do protocolo MDB.

Neste projeto será apresentado:

- **Alterações no Escopo do Projeto**(p.4), mostra as alterações realizadas durante o desenvolvimento para que o projeto pudesse atender ao seu objetivo primário;
- **Detalhamento Físico e Tecnologias Utilizadas**(p.6), apresenta como é usado cada módulo do sistema, e como eles atuam;
- **Central de Telemetria**(p.8),mostra quais as operações básicas, e explica como elas ocorrem internamente ao sistema;
- **Servidor**(p.18), mostra as classes e diagramas de relacionamento para o modelo de banco de dados do sistema;
- **Funcionamento do Servidor**(p.22), informações sobre a sequência de comandos executados pelo servidor;
- **Emulador MDB** (p.31), mostra como funciona o emulador desenvolvido pela equipe;
- **Tecnologias utilizadas** (p.31), mostra os módulos de hardware necessários para o funcionamento do sistema;
- **Procedimentos de Teste e Validação**(p.49), métodos aplicados para teste do sistema desenvolvido;
- **Análise de Riscos** (p.54), apresenta possíveis riscos previstos no desenvolvimento do projeto;
- **Viabilidade e Conclusões**(p.57), traz o estudo feito para levantar a viabilidade do projeto, face às dificuldades e concorrentes. As conclusões trarão as considerações finais sobre o projeto;
- **Cronograma**(p.59), detalha as datas previstas para a finalização de cada etapa do projeto.

Alterações no Escopo do Projeto

No decorrer do desenvolvimento do protótipo e fase final foram observadas dificuldades que exigiram alterações no Escopo do Projeto. Em outros casos foram observadas que as funcionalidades propostas estariam em desacordo com a proposta de “Telemetria”, que visa a coleta e registro de dados a serem disponibilizados para processamento.

> Definição: **Telemetria**
>
> *Registro de dados à distância obtidos através de instrumentos de medida.*
>
> ***Exemplos:***
>
> *Telemetria é uma tecnologia que permite a medição e comunicação de informações de interesse do operador ou desenvolvedor de sistemas. A palavra é de*

origem Grega onde tele = remoto e metron = medida. Sistemas que necessitam de instruções e dados enviados a eles para que sejam operados, requerem o correspondente a telemetria, o telecomando.

É um sistema de monitoramento com diversas aplicações, muito falada nas corridas como Fórmula 1, Dragsters e qualquer outro tipo de esporte automobilístico, também é muito usada em indústrias de monitoramento, normalmente funciona via transmissão sem fio (sinal de rádio), daí o nome telemetria. Em muitos lugares tem o uso em conjunto do Datalog que é a função de gravar um período de tempo da leitura dos canais da telemetria. O sistema também é utilizado para recolhimento de dados meteorológicos.

FONTE: Wikipédia – A Enciclopédia Livre, http://pt.wikipedia.org/wiki/Telemetria

Por este motivo, a fim de garantir a conclusão dentro do prazo, foram eliminados do escopo os seguintes itens:

- **Controle do Prazo de Validade:** A implementação de um controle de prazo de validade para os itens inseridos na máquina implicariam num possível atraso no desenvolvimento de outras funcionalidades. Haveria uma complexidade adicional por parte do servidor na gerência dos estoques do depósito e de cada máquina.
- **WebService:** O servidor está sendo implementado de modo a permitir o envio de consultas via POST por terceiros. O objetivo é reaproveitar funcionalidades desenvolvidas para o funcionamento remoto das centrais de telemetria, sem haver a necessidade de implementar um servidor de serviços dedicado.
- **Instalação e Apresentação:** Em função da dificuldade de se encontrar uma máquina onde a central de telemetria pudesse ser instalada para a apresentação final, considera-se que está será realizada de forma simulada conforme manuais descritivos do MDB através de *hyperteminal*.
- **Máquina Interativa:** O módulo a ser apresentado não terá suporte a sistema interativo com dados nutricionais e interação avançada com o usuário, devido a falta de tempo para o desenvolvimento de execução de tal atividade.
- **Alertas personalizados:** Os alertas personalizados não serão implementados por não fazerem sentido a real aplicação do projeto.
- **Interpretação de Arquivos MDB/DTS:** Os arquivos MDB/DTS não serão interpretados devido a desviar o escopo inicial traçado.
- **Mais de uma operação por transmissão:**Não houve necessidade de condensar múltiplas operações em uma única transmissão para economia de tráfego.
- **Atualização de preços na VM:** Não houve um entendimento genérico do procedimento de atualização de preços, uma vez que não poderia ser realizado através do barramento MDB.
- **Alertas customizados:** Não se fez necessidade da configuração deste tipo de alerta, levando em conta que nenhum exemplo deste tipo de necessidade foi encontrado.

- **Mensagens *Keep-Alive* Tabelas de IP:** Foi desconsiderada a implementação destas mensagens,pois não se observou a necessidade de que o servidor inicie comunicação com as Centrais de Telemetria, uma vez que servidor é única e exclusivamente responsável por responder as requisições geradas.
- **Validação de dados:** Os dados inseridos nos formulários não foram validados, devido a falta de necessidade perante o foco principal do projeto e utilização do módulo de telemetria.
- **Histórico de vendas com periodicidade:** Não foi implementado devido a falta de necessidade para com a empresa administradora de VM's, tornando-se fora do escopo do projeto. Entendemos que o objetivo do projeto é reunir os dados e disponibilizá-los para terceiros, de modo que essa funcionalidade poderá ser explorada pelos integradores.

Detalhamento Físico e Tecnologias Utilizadas

O módulo G24 é um modem *Quadriband*, homologado pela Anatel, com conexão direta a internet via GPRS. Este possui duas portas seriais e suporte a USB 2.0, possuindo uma porta do protocolo. O processador, G24, é programado em Java e de tamanho reduzido. Ele conta com 1GB de memória RAM e 10MB de memória FLASH (maiores informações sobre o modem página 29 deste documento).

O diagrama abaixo apresenta de forma resumida o aspecto organizacional do sistema implementado. Temos separados como duas entidades distintas a VM e o servidor remoto com que o módulo de telemetria se comunica.

O módulo de telemetria, que foi o foco deste projeto, é responsável por ler as informações da VM e fazer a comunicação via GPRS com o servidor. Este então mantém um registro detalhado de clientes (empresas que utilizam o sistema), identificação das máquinas, disponibilidade de itens, etc.

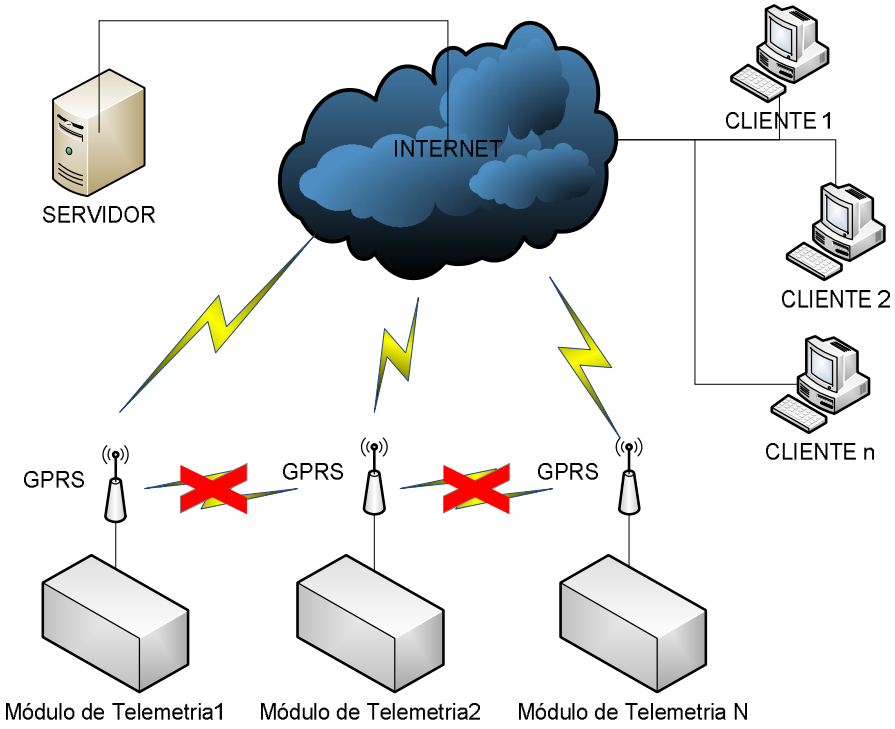


Figura 1: Visão organizacional da comunicação

A imagem abaixo apresenta uma noção melhor das operações que podem ser realizadas na VM. Observe que o módulo de telemetria apenas se comunica com o servidor, transmitindo o resultado da operação realizada pelo usuário da máquina. Nenhuma outra entidade além do servidor pode se comunicar diretamente com o módulo.

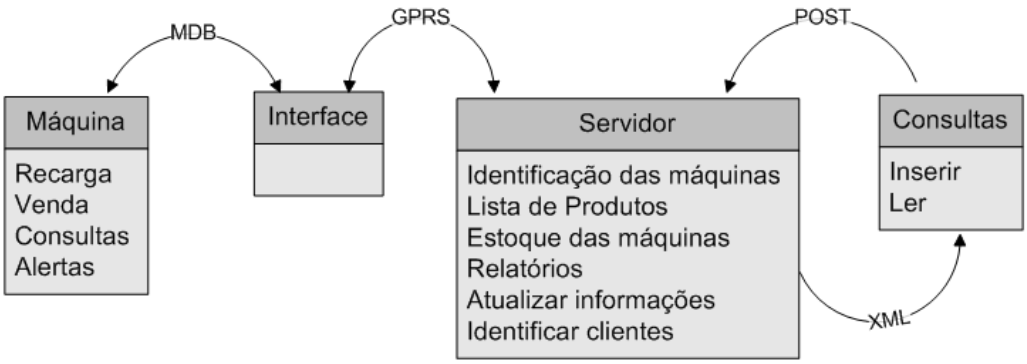


Figura 2: Visão organizacional do sistema

Observe que há uma entidade adicional externa capaz de realizar consultas e inserções no servidor através de comandos HTTP/POST. Esta abordagem permite que o software rodando no servidor seja facilmente “anexado” a outros aplicativos de gestão comercial que existem no mercado. A troca de arquivos XML será feita via Internet, permitindo que as informações estejam disponíveis online em qualquer lugar.

Deste modo também se permite que o sistema seja independente de interface gráfica. Todas as operações no servidor podem ser executadas a partir de scripts que interpretam os comandos enviados. O tratamento das informações coletadas então é completamente independente do nosso servidor, e fica por conta das empresas responsáveis por desenvolver sistemas de gestão comercial. O servidor também oferece uma interface simplificada para consultas e inserções.

1. Central de Telemetria

O módulo de telemetria, responsável pela comunicação com o servidor, é capaz de realizar algumas operações básicas: Registro de Vendas, Envio de Alertas, e Registro de Recarga dos Cartuchos, descritas mais adiante. Para o registro das vendas, fazemos uso do protocolo MDB.

> Definição: **Protocolo MDB**
>
> É um tipo de comunicação utilizada entre a VMC (Controlador de *Vending Machine*) e os periféricos da *Vending Machine*. Existem vários tipos de periféricos, além de vários níveis de operação de periféricos. Quando são de níveis mais altos tem maior complexidade de resposta e operação.
>
> Maiores detalhes na Seção **Tecnologias utilizadas**, subseção 3 - Barramento MDB (Multi-Drop Bus)

Também conta com um display LCD, onde o operador da máquina pode visualizar diversas informações, o *menu* principal, e o retorno de algumas funções, como a visualização do estoque, por exemplo.

Como o projeto de uma *Vending Machine* não está presente no escopo do projeto, ela é simulada através de sinais injetados no módulo. Assim deseja-se na apresentação ter um protótipo funcional que possa simular uma VM, e que seja capaz de demonstrar o correto funcionamento do sistema de telemetria desenvolvido.

Para isso, usamos um emulador do barramento MDB. Com este, podemos ajustar cada byte que desejemos, e então podemos enviá-lo à central de telemetria. O emulador está descrito na seção 3. Emulador MDB, e foi desenvolvido pela equipe para uso no projeto.

Uma vez interpretada à operação realizada no barramento, ou então iniciada uma ação pelo operador da central. O módulo de telemetria abre uma conexão com o servidor remoto através da seguinte função, em Java:

```
static String net(String parametro, String ip, String url) throws
IOException
```

Esta função está na classe ServicosHttp, do programa que é executado no G24 (firmware). Como se vê, ela possui três parâmetros, dois dos quais dificilmente se alteram nas várias chamadas a esta função.

O primeiro informa a identificação da central e será enviado como um POST ao servidor, em algumas operações pode conter mais informações para que a função solicitada seja executada. Para reduzir a alteração de código quando da mudança deste parâmetro, temos uma variável estática, na classe *menu*, que contêm o nome da central. Este parâmetro nunca se altera, para uma mesma central, pois é referenciado pelo número IMEI (Identificação Internacional de Equipamento Móvel), já que este trata-se de um número único para cada equipamento de telefonia, neste caso o modem.

O segundo parâmetro corresponde ao IP do servidor, que pode sofrer alteração em função de possuir IP dinâmico. Raramente se altera, mas é buscado novamente cada vez que o operador entra no *menu* da máquina.No projeto fazemos uso do domínio *HTTP://tcc.gateon.com.br/*como uma ferramenta de Proxy para localização do IP do servidor responsável por armazenar e processar as informações de telemetria.

O último parâmetro identifica a função que se deseja executar, portanto é modificado para praticamente cada requisição.

Mais detalhes sobre esta função estão na seção 2.3 - Conexão com a Internet, na página 45.

Em todas as descrições futuras dos procedimentos a serem realizados para transmissão dos dados, as operações deverão ser realizadas pelo módulo de telemetria do mesmo modo como apresentado acima.

As operações de conexão e transmissão das URL's solicitadas também são realizadas pelo módulo e sempre com o mesmo servidor especificado, que por sua vez, responde com uma confirmação.

O envio de informações ocorre através da mensagem na conexão, via POST e nunca na URL (GET).

As funções estarão exibidas aqui, na mesma ordem em que aparecem no *menu* da central de telemetria. Para maiores informações sobre as funções aqui listadas, consulte o Anexo 1: Firmware do Modem G24-JAVA.

1.1. Status

Verifica o Status da máquina junto ao servidor. Este status pode ser **Não inicializada**, ou então informa a situação do conteúdo da máquina. Apenas máquinas inicializadas podem operar com o servidor, gerar logs, atender ordens de serviço, etc. A inicialização deve ser o primeiro procedimento a ser executado em uma central de telemetria.

O retorno será uma mensagem tipo texto com formatação XML, conforme exemplos apresentados abaixo:

Para uma central não inicializada

```
<?xmlversion="1.0" encoding="ISO8859-1" ?>
<Conteudo>
   <Alertas>
      <erro>
        <ID>1</ID>
        <Mensagem>Central não Inicializada</Mensagem>
      </erro>
    </Alertas>
</Conteudo>
```

Para uma central inicializada

```
<?xmlversion="1.0" encoding="ISO8859-1" ?>
<Conteudo>
   <Alertas/>
   <Sucesso>
      <Mensagem>Situação do Estoque Registrado no Servidor</Mensagem>
        <Data>2010-11-26 12:09:31</Data>
        <nCartuchos>10</nCartuchos>
        <Cartuchos>
          <Cartucho>
            <Numero>1</Numero>
            <idProduto>1</idProduto>
            <Estoque>
              <Capacidade>50</Capacidade>
              <Atual>0</Atual>
              <Minimo>20</Minimo>
            </Estoque>
            <Detalhes>
              <Nome>produtox Teste 2</Nome>
              <Preco>14.00</Preco>
              <Descricao>Uma descrição de um produto</Descricao>
            </Detalhes>
          </Cartucho>
      </Cartuchos>
   </Sucesso>
</Conteudo>
```

Os campos deste XML são auto explicativos e se referem ao conteúdo presente em cada Cartucho da *Vending Machine*. Entende-se como sendo um cartucho a estrutura interna da máquina responsável por armazenar um conjunto de produtos do mesmo tipo.

A contagem de elementos do tipo *Cartucho* pode variar de *1..nCartuchos*, dependendo do número de cartuchos inicializados no servidor. Se um cartucho estiver registrado como vazio, ele não aparece na resposta.

Embora esteja presente o campo *Detalhes->Descricao* dentro de cada estrutura do tipo *Cartucho*, esta informação não é utilizada pela central de telemetria. Este campo foi mantido como proposta de que mais informações poderiam ser trocadas entre servidor e central, a fim de permitir a criação de *Vending Machines* mais interativas.

Parâmetros utilizados:

Aqui descreveremos os parâmetros para a já citada função "net".

```
ServicosHttp.net("CentralTelemetria=" + Menu.nome_central,Menu.ip,
"status")
```

A função utilizada é "status".

1.2. Inicializar

Inicializa a máquina junto ao servidor. Após esta operação, a máquina estará habilitada a executar as demais funções do sistema.

Retorna um XML indicando se conseguiu inicializar, ou se algum erro foi encontrado.

Parâmetros utilizados:

```
ServicosHttp.net("CentralTelemetria="     +     Menu.nome_central,
Menu.ip, "inicializar")
```

Retorno para uma central não inicializada (em caso de sucesso)

```
<?xmlversion="1.0" encoding="ISO8859-1" ?>
<Conteudo>
  <Alertas>
    <erro>
      <ID>1</ID>
      <Mensagem>Central não Inicializada</Mensagem>
    </erro>
  </Alertas>
  <Sucesso>
    <Mensagem>Central Inicializada com Sucesso!</Mensagem>
    <DataHora>2010-11-26 12:31:33</DataHora>
    <nCartuchos>5</nCartuchos>
    <Cartuchos>
      <Cartucho>
      -- Conforme apresentado em Status
      </Cartucho>
    </Cartuchos>
  </Sucesso>
</Conteudo>
```

A contagem de elementos do tipo *Cartucho* pode variar de *1..nCartuchos*, dependendo do número de cartuchos inicializados no servidor. Se um cartucho estiver registrado como vazio, ele não aparece na resposta.

O campo *nCartuchos* é utilizado durante a inicialização da máquina a fim de permitir a gravação do endereço de cada cartucho, conforme especificado pelo barramento MDB. Desta forma a máquina é inicializada uma única vez.

A situação de *Estoque->Atual* é sempre inicializada como zero. O valor sofrera alteração após a execução da primeira ordem de serviço criada.

Retorno para uma central inicializada

```
<?xmlversion="1.0" encoding="ISO8859-1" ?>
<Conteudo>
  <Alertas>
    <erro>
      <ID>4</ID>
      <Mensagem>Central já Inicializada</Mensagem>
    </erro>
  </Alertas>
  <Sucesso>
    <Mensagem>Situação do Estoque Registrado no Servidor</Mensagem>
    <Data>2010-11-26 12:32:41</Data>
    <nCartuchos>10</nCartuchos>
     <Cartuchos>
      -- Conforme apresentado em Status
    </Cartuchos>
  </Sucesso>
</Conteudo>
```

1.3. Estoque

Verifica, no servidor, o que consta como estoque atual da máquina. Pode ser utilizada quando o operador deseja comparar os dados do servidor, com a quantidade de itens realmente presentes na máquina.

Retorna os dados de estoque, em um XML conforme apresentado na operação **Status**. Estes dados são exibidos no visor LCD.

Parâmetros utilizados:

```
ServicosHttp.net("CentralTelemetria="  +  Menu.nome_central, Menu.ip, "status");
```

1.4. Ordens de Serviço

Recupera as ordens de serviço destinadas a esta central de telemetria. O operador pode então escolher qual ordem executar, bem como detalhar cada ordem, verificando seus itens e quantidades.

Retorna um XML, contendo a descrição de cada ordem de serviço.

Parâmetros utilizados:

```
ServicosHttp.net("CentralTelemetria=" + Menu.nome_central,Menu.ip,
"os");
```

Exemplo de retorno

```
<?xmlversion="1.0" encoding="ISO8859-1" ?>
<Conteudo>
  <Alertas/>
  <Sucesso>
    <Mensagem>Resumo das Ordens de Servico</Mensagem>
    <OrdensDeServico>
      <OS>  -- 1..n registros
        <Data>2010-11-02 09:01:23</Data>
        <idOrdemServico>25</idOrdemServico>
        <Produtos>
          <Produto> -- 1..n registros
          <idProduto>1</idProduto>
          <Cartucho>1</Cartucho>
          <Nome>produtox Teste 2</Nome>
          <Quantidade>10</Quantidade>
          </Produto>
        </Produtos>
      </OS>
    </OrdensDeServico>
  </Sucesso>
</Conteudo>
```

1.5. Recarga

O processo de recarga depende de uma ordem de serviço emitida pelo servidor. Esta contem uma listagem de itens mínima que deverá ser levada pelo funcionário de reabastecimento a fim de completar o estoque da máquina.

Uma vez que todos os cartuchos de produtos tenham sido preenchidos, um comando presente no *menu* do módulo dispara um aviso ao servidor de que a ordem de serviço foi realizada. Assumindo que entre a emissão da ordem de serviço e o ato de recarga tenham sido realizadas novas vendas, o funcionário responsável por reabastecer a máquina tem a opção de completar todas as posições vagas dos cartuchos. Tendo reabastecido 100% a máquina, a opção completar no *menu* de ordem de serviço deve ser acionada, informando ao servidor de que a VM está totalmente reabastecida.

O display LCD permite ao operador da máquina observar qual a ordem de serviço deve ser executada, além de detalhar a quantidade de itens presente em cada cartucho naquele

instante. Esta operação permite ao operador ou a um auditor observar a ocorrência de furtos ou falhas de comunicação.

Parâmetros utilizados:

```
ServicosHttp.net("CentralTelemetria=" + Menu.nome_central+
“idOrdemServico” + ...,Menu.ip,"completa");
```

O primeiro parâmetro nesta função não é somente o id da central, mas também o id da ordem de serviço a ser executada.

Exemplo retorno

```
<?xmlversion="1.0" encoding="ISO8859-1" ?>
<Conteudo>
    <Alertas/>
    <Sucesso>
      <Mensagem>Ordem de Serviço atendida com sucesso!</Mensagem>
       <DataHora>2010-11-26 12:59:11</DataHora>
    </Sucesso>
</Conteudo>
```

A Figura 3 apresenta o diagrama com as possíveis ações de recarga.

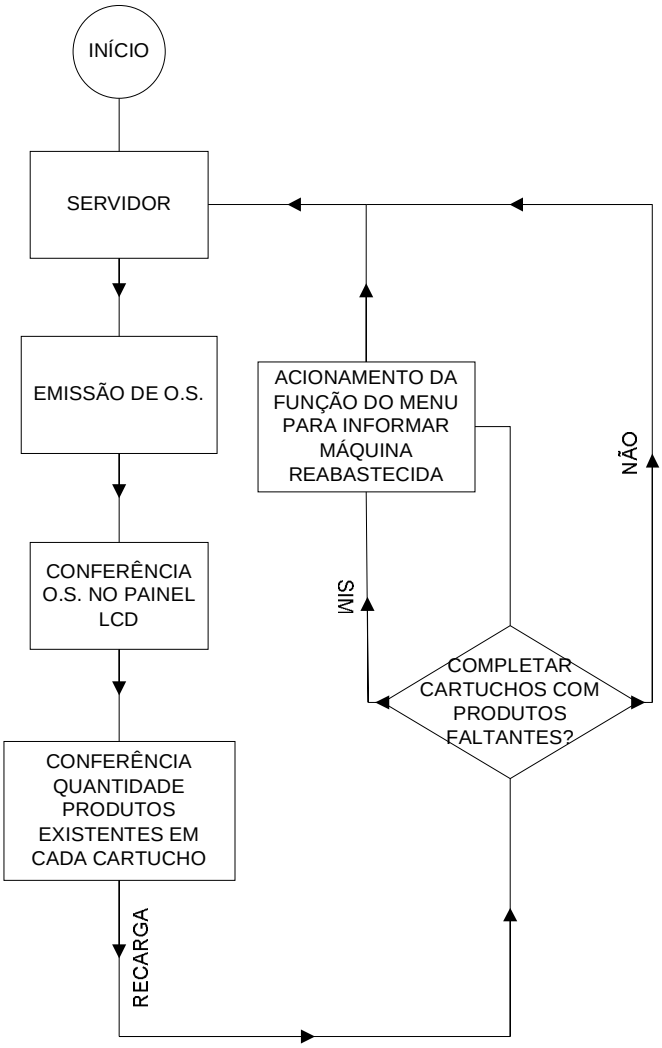


Figura 3 - Procedimento de Recarga

Parâmetros utilizados:

```
ServicosHttp.net("CentralTelemetria=" + Menu.nome_central,
Menu.ip, "completa");
```

A função utilizada no servidor é “completa”.

Exemplo de retorno

```
<?xmlversion="1.0" encoding="ISO8859-1" ?>
<Conteudo>
  <Alertas/>
  <Sucesso>
     <Mensagem>Máquina abastecida por completo!</Mensagem>
     <DataHora>2010-11-26 01:04:44</DataHora>
   </Sucesso>
```

```
   <Sucesso>
      <Mensagem>Situação do Estoque Registrado no Servidor</Mensagem>
      <Data>2010-11-26 01:04:44</Data>
      <nCartuchos>10</nCartuchos>
       <Cartuchos>
          -- Conforme apresentado em Status
      </Cartuchos>
    </Sucesso>
</Conteudo>
```

1.6. Alertas

O módulo de telemetria é capaz de receber informações de sensores espalhados pela VM a fim de comunicar ao servidor possíveis operações ilegais.

Por exemplo: a máquina pode contar com acelerômetros que monitoram tentativas de furto; O sistema pode ser dotado de bateria para comunicar ao servidor quedas de energia; Se a máquina for sensoriada, pode acusar falha na entrega de um item, ou de que o reservatório de dinheiro está cheio.

Para o sensoriamento, o módulo de telemetria é dotado de 11 pinos de comunicação paralela. Quando algum dado for inserido nas portas para sensores, o sistema gera um pulso no pino de interrupção do módulo. Assim, esta envia para o servidor a situação atual dos sensores.

Procedimento de aquisição de dados da *Vending Machine*:

Conectar os sensores da máquina nas portas disponíveis no módulo de telemetria. Os sensores devem ter saída digital TTL. O servidor é avisado quando qualquer um deles trocar de estado (5v -> 0v ou 0v -> 5).

Parâmetros utilizados:

```
ServicosHttp.net("CentralTelemetria="+Menu.nome_central + "&XML="
+ alerta, Menu.ip, "alerta");
```

Neste caso, o primeiro parâmetro carrega também um XML, contendo os dados sobre os sensores que foram ativados, na variável “alerta”.

Exemplo do XML transmitido

```
<Conteudo>
  <Alertas>
    <Alerta>
      <Mensagem>Teste</Mensagem>
      <Sensor>1</Sensor>
      <Estado>TRUE</Estado>
      <DataHora>2010-11-26 01:14:40</DataHora>
    </Alerta>
  </Alertas>
```

```
</Conteudo>
```

Exemplo de retorno

```
<?xmlversion="1.0" encoding="ISO8859-1" ?>
<Conteudo>
    <Alertas/>
    <Sucesso>
      <Alerta>
        <Mensagem>Sensor 1 registrado com sucesso. (Teste)</Mensagem>
        <ID>97</ID>
      </Alerta>
    </Sucesso>
</Conteudo>
```

1.7. Vendas

O processo de venda ocorre quando um produto é vendido pela *Vending Machine.* Para este processo a máquina conta com uma porta do protocolo MDB, que detecta quando um produto foi vendido pela máquina. Na inicialização da máquina, os códigos MDB dos cartuchos já foram definidos, portanto na operação de venda, o servidor só é informado de qual produto foi vendido para que a baixa seja feita no estoque do servidor.

Para informar a venda de um produto, um XML é enviado contendo dados como, número do cartucho ao qual o produto foi consumido, id do produto, preço e data/hora da venda. Através deste XML o servidor, diminui o produto do estoque e responde com um XML, confirmando o processo de venda.

Exemplo de XML de uma venda:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Vendas>
   <Produto>
      <idXML>indice</idXML>
      <idProduto>p2.id_produto</idProduto>
      <Cartucho>p2.cartucho</Cartucho>
      <Preco>p2.preco</Preco>
      <DataHora>1988/06/03 12:00:01</DataHora>
   </Produto>
</Vendas>
```

Exemplo de resposta do servidor:

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<Conteudo>
   <Alertas />
      <Sucesso>
      <Vendas>
```

```
        <idXML>16</idXML>
      </Vendas>
    </Sucesso>
</Conteudo>
```

1. Servidor

A implementação do servidor foi feita em PHP, facilitando o acesso ao banco de dados e a realização de operações via browser.

1.1. Identificação do Cliente

Entende-se por cliente a empresa administradora de VM's cadastrada no sistema. Uma empresa pode ter uma ou mais VM's utilizando os serviços.

1.2. Identificação da Máquina

Cada VM deve ser identificada pela sua localização (endereço). Há a necessidade de identificar o número de série do módulo de telemetria que está conectada a ela, pois é o identificador da máquina.

A VM obrigatoriamente deve estar relacionada a um cliente, assim como informar o número de cartuchos disponíveis para itens diferentes. Há a possibilidade de incluir um nome para a máquina.

Figura 4: Identificação do Cliente

Figura 5: Identificação da Máquina

1.3. Estoque de Produtos

Cada cliente deve catalogar no sistema os produtos vendidos em suas máquinas. A identificação do fornecedor é opcional.

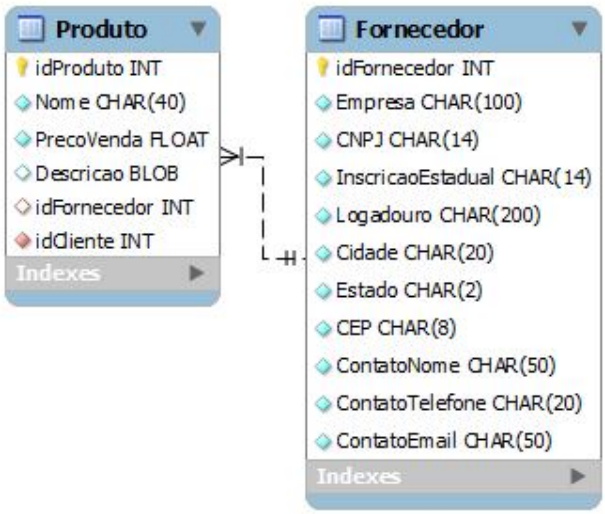


Figura 6: Estoque de Produtos

1.4. Estoque das Máquinas

Este conjunto de informações relaciona os produtos do estoque com o cartucho de cada VM do cliente. Através desta classe de informações é possível identificar a capacidade de cada

cartucho em função do item presente nele, além de se saber quantos itens ainda estão disponíveis no interior das máquinas.

Tais informações permitem a emissão de ordens de serviço, incluindo a possibilidade de renovação de estoque dos itens.

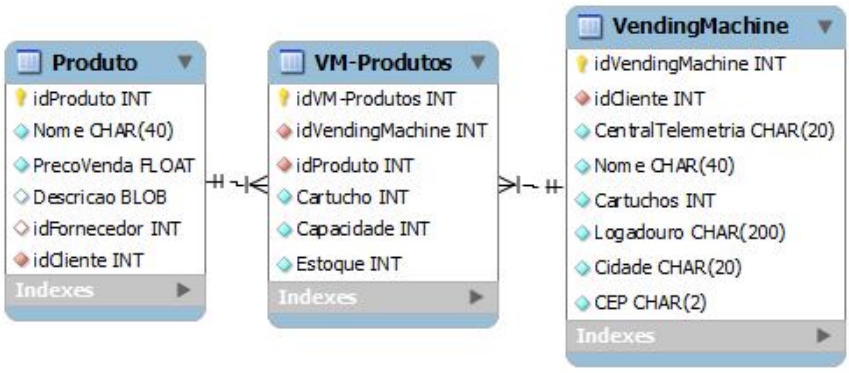


Figura 7: Estoque das Máquinas

1.5. Relatórios

Ainda no servidor é possível emitir um conjunto de relatórios com base nas informações cadastradas no sistema. Estes podem ser consultados remotamente via requisição, a fim de permitir a atualização do banco de dados da administradora conforme a integração realizada ao sistema de gestão comercial utilizado pela empresa. Os relatórios são:

1. Ordens de Serviço para Reabastecimento
2. Histórico de Vendas
3. Histórico de Alertas
4. Ordens de Auditoria

O último relatório é um tipo especial de ordem de serviço, permitindo que uma equipe diferente cheque as máquinas verificando possíveis furtos de itens, uma vez que ele informa a quantidade de itens presente na VM.

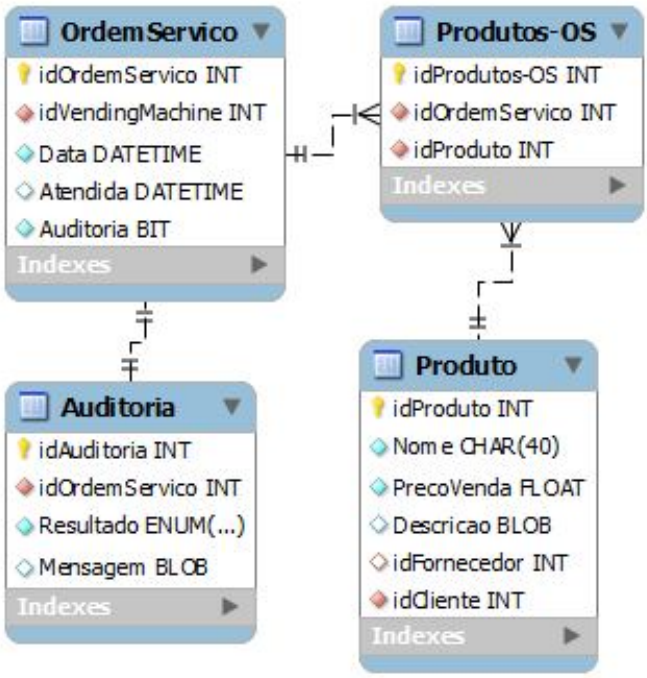


Figura 8: Registro de Ordens de Serviço

Adicionalmente todo alerta recebido pelo servidor é imediatamente enviado ao endereço de e-mail do cliente registrado. Esta medida visa que a administradora da VM possa iniciar uma ação corretiva o mais breve possível, mesmo que a empresa não faça uso de um software de gestão comercial.

Figura 9: Registro de Alertas

1.6. Atualização de Informações

Outro conjunto de operações permite atualizar a descrição das máquinas, atualizar estoques do armazém e inserir dados de furtos.

Note que não há possibilidade de editar diretamente o estoque de uma máquina, pois esta operação exige uma ordem de serviço. Deste modo editando a descrição da máquina é possível substituir um cartucho de itens por outro de outro tipo. Este tipo operação é declarado na próxima ordem de serviço.

1.7. Troca de Arquivos

Toda a interface com softwares terceiros é feito a partir de requisições com dados enviados via POST. O servidor gera um novo arquivo XML com a resposta desejada (se necessário).

As operações disponíveis, assim como o processo de troca de mensagens, serão documentadas futuramente no manual de integração.

2. Funcionamento do Servidor

A ferramenta de desenvolvimento utilizada para os scripts PHP será o NetBeans 8.0. Para configuração e implementação do servidor foi feito uso dos seguintes componentes:

- Computador PC com:
    - Windows Server 2003
    - WampServer (Apache + PHP 5 + MySQL)

O WampServer é um conjunto de aplicativos voltados a servidores WEB pré-configurado, facilitando a instalação e a manutenção dos componentes.Foram utilizadas suas configurações padrão.

A única exigência para a hospedagem externa do servidor é o suporte a PHP 5 e MySQL. O ambiente operacional não é importante. Como o host contratado não oferecia suporte ao PHP 5.3, o servidor com IP fixo atua apenas como um Proxy informando o endereço do nosso servidor de desenvolvimento. Este é um servidor próprio que está sempre conectado à Internet, porém não conta com IP fixo.

Para as mensagens recebidas do módulo de telemetria, o servidor funciona através de requisições de páginas/scripts PHP através do protocolo HTTP. Todos os parâmetros para execução do script são enviados via comandos GET e POST.

De maneira geral pode-se descrever o funcionamento básico dos scripts da seguinte forma:

1. Copiar os parâmetros enviados para variáveis de memória;
2. Iniciar o processo adequado de atualização do banco de dados
3. Retorna “OK”, ou se necessário, retorna arquivo XML com as informações requisitadas.

De forma sucinta, a seguir são descritas as operações básicas de interface com o usuário.

2.1. Cadastro de Usuários e Login

Arquivo: Index.php

1. Solicitar usuário e senha para acesso ao *menu*;
    a. Se o usuário e senha estiverem corretos, redireciona para Menu.php;
2. Oferece possibilidade de recuperar usuário e senha
    a. Se o e-mail existir no banco de dados, envia as informações de acesso;

b. Caso contrário, informa erro;

3. Oferece cadastro de novo usuário
    a. Solicita dados para novo cadastro;
    b. Insere usuário no banco de dados;
    c. Envia e-mail de confirmação para o usuário;
    d. Aguarda validação de e-mail;
    e. Redireciona para Menu.php;

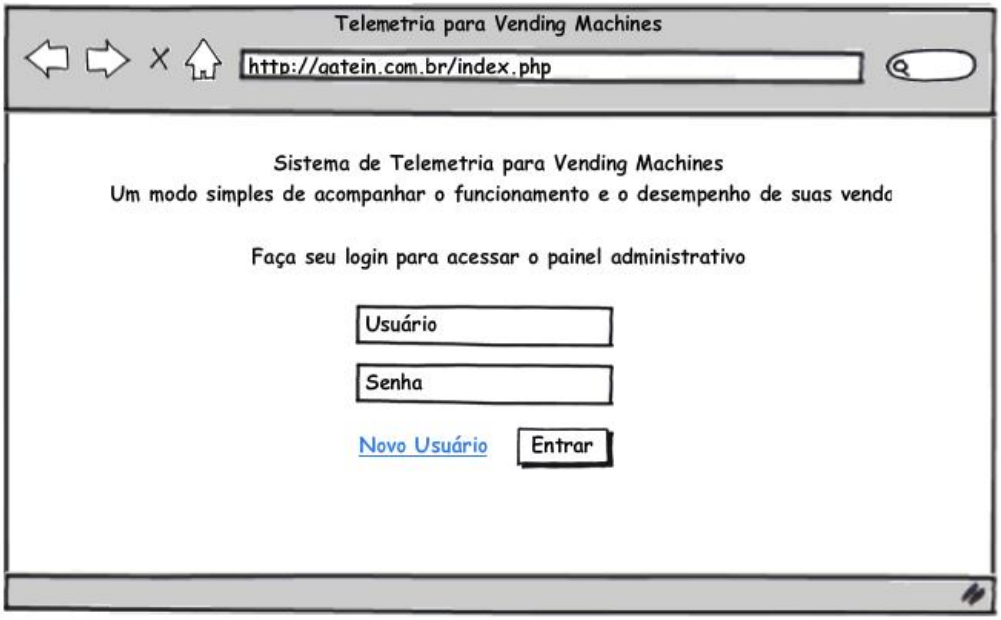

Telemetria para Vending Machines
http://gatein.com.br/index.php
Sistema de Telemetria para Vending Machines
Um modo simples de acompanhar o funcionamento e o desempenho de suas vend
Faça seu login para acessar o painel administrativo
Usuário
Senha
Novo Usuário
Entrar

Figura 10: Esboço - Tela de Login

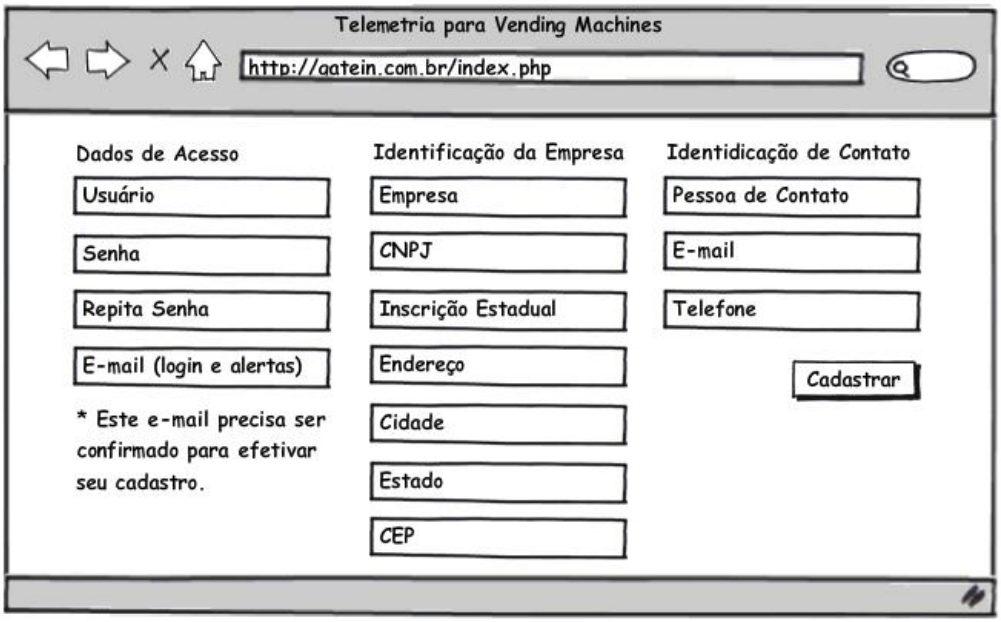


Figura 11: Esboço - Tela de Cadastro

2.2. Menu de operações básicas

Arquivo: Menu.php

1. Exibe sequencia de *menus*;
    a. Cadastro de *Vending Machines*
        i. Redireciona para: NovaMaquina.php
    b. Cadastro de Produtos
        i. Redireciona para: ProdutosCadastro.php
    c. Estoque das *Vending Machines*
        i. Redireciona para: ProdutosEstoqueVM.php
    d. Relatórios
        i. Redireciona para: Relatorios.php
    e. Atualizar Dados
        i. Redireciona para: AtualizarDados.php
    f. Envio de Arquivos
        i. Redireciona para: Arquivos.php

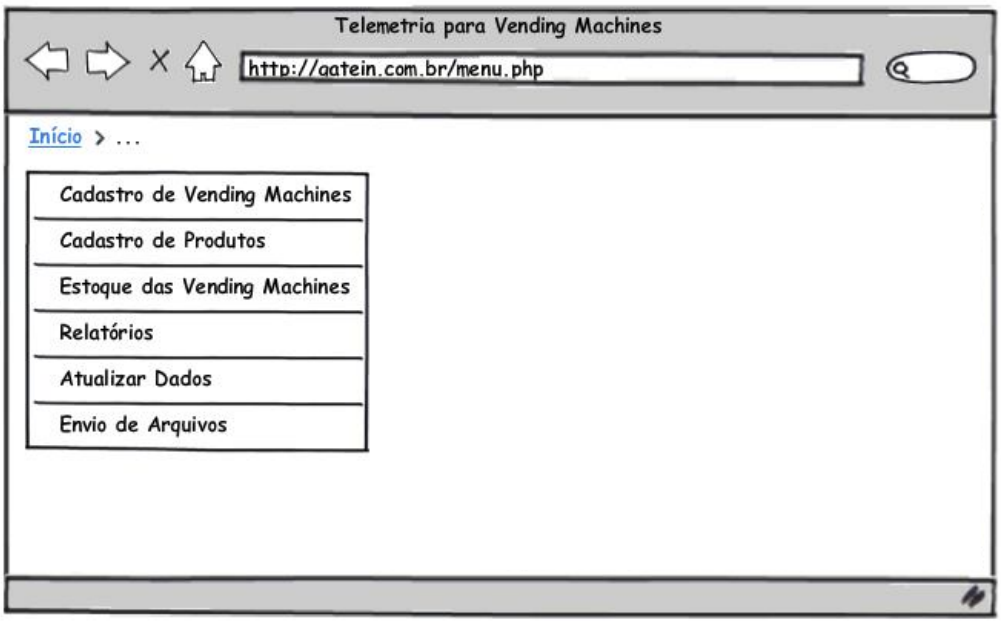


Figura 12: Esboço - Tela Inicial

2.3. Cadastro de *Vending Machines*

Arquivo: NovaMaquina.php

1. Oferece formulário para cadastro de nova *Vending Machine*;
4. Insere registro no banco de dados;
5. Envia confirmação de registro para o e-mail cadastrado;
6. Aguarda inicialização da máquina;
7. Envia confirmação de inicialização para o e-mail cadastrado;

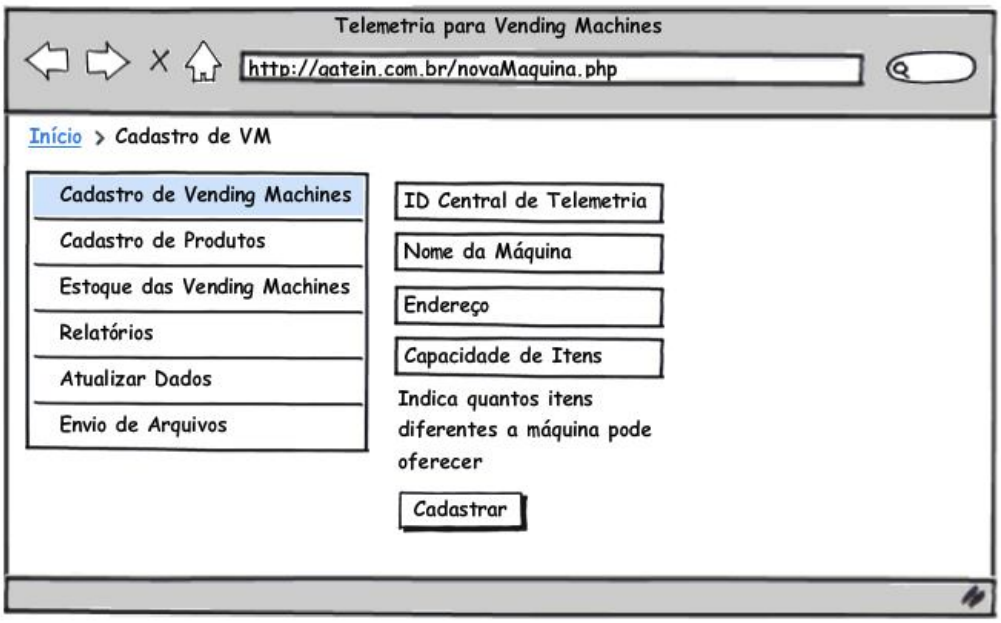


Figura 13: Esboço - Tela de Cadastro de *Vending Machines*

2.4. Cadastro de Produtos

Arquivo: ProdutosCadastro.php

1. Oferece formulário para cadastro de um novo produto;
2. Insere registro no banco de dados.

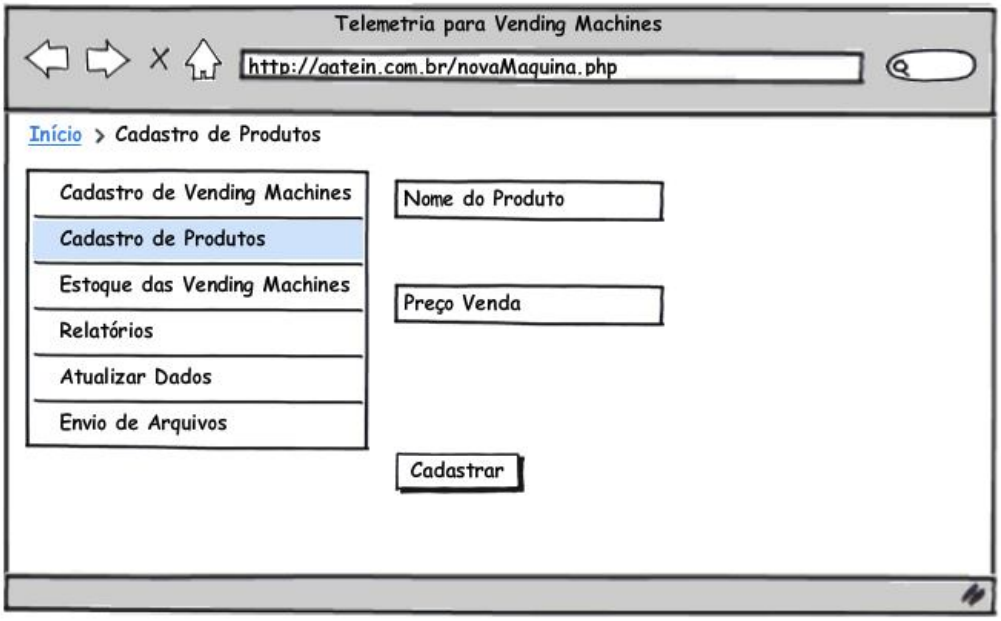


Figura 14: Esboço - Tela de Cadastro de Produtos

2.5. Registro de estoque das *Vending Machines*

Arquivo: ProdutosEstoqueVM.php

1. Verifica no banco de dados as *Vending Machines* do usuário;
2. Verifica no banco de dados os produtos do usuário;
3. Oferece formulário para configuração do estoque das máquinas;
4. Insere registro no banco de dados;
5. Emite ordem de serviço para inicializar a máquina;
6. Envia ordem de serviço para o e-mail registrado.

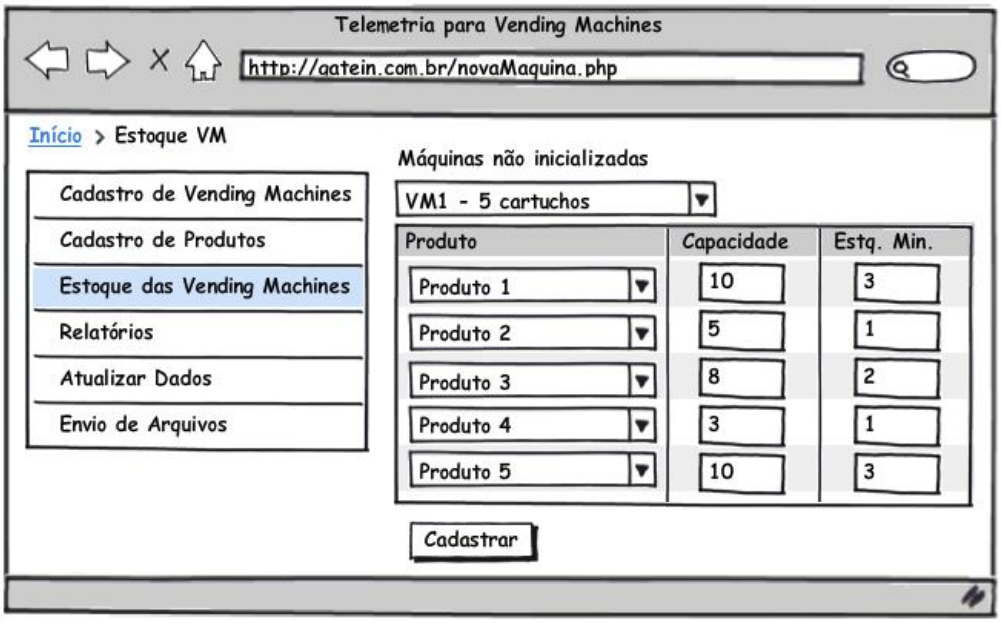


Figura 15: Esboço - Tela de Relacionamento de Produtos

2.6. Emissão de Relatórios

Arquivo: Relatorios.php

1. Oferece opções de relatório:
    a. Ordens de Serviço[1];
        i. Oferece possibilidade de emitir OS com base em:
            1. Regras automáticas de reabastecimento;
            2. Reabastecer Tudo;
            3. Seleção Manual de *Vending Machines*;
            4. Seleção Manual de Produtos;
        ii. Exibe ordens de serviço para impressão e/ou envia por email;
    b. Histórico de Vendas
        i. Permite ver um resumo das vendas da máquina;
    c. Histórico de Alertas[2];
        i. Permite visualizar os registros de alertas recebidos;
    d. Ordens de Auditoria;

i. Permite emitir ordens de auditoria para:
    1. Todas as máquinas;
    2. Percentual de máquinas;

[1]Os relatórios de OS são enviados automaticamente ao e-mail cadastrado quando as regras automáticas são atendidas.

[2]O administrador é informado automaticamente via e-mail sempre que um sensor for ativado na *Vending Machine*.

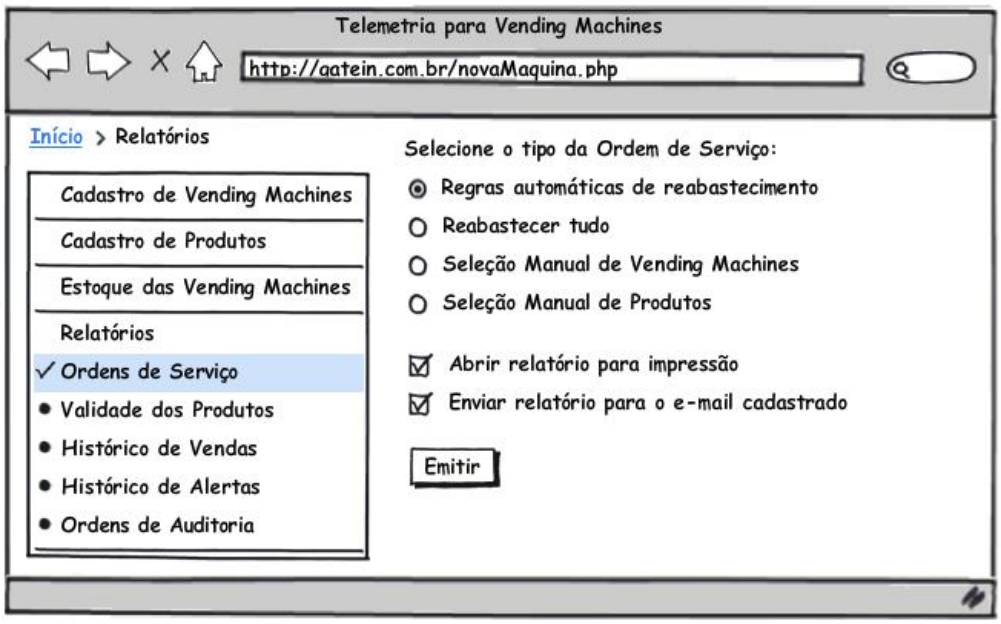


Figura 16: Esboço - Tela de Emissão de Relatórios

2.7. Atualização de Informações

Arquivo: AtualizarDados.php

1. Oferece opções para Atualizar:
    a. Dados cadastrais
    b. Lista de produtos
    c. Informações e Estoque das *Vending Machines*

As operações fazem uso das mesmas rotinas já descritas.

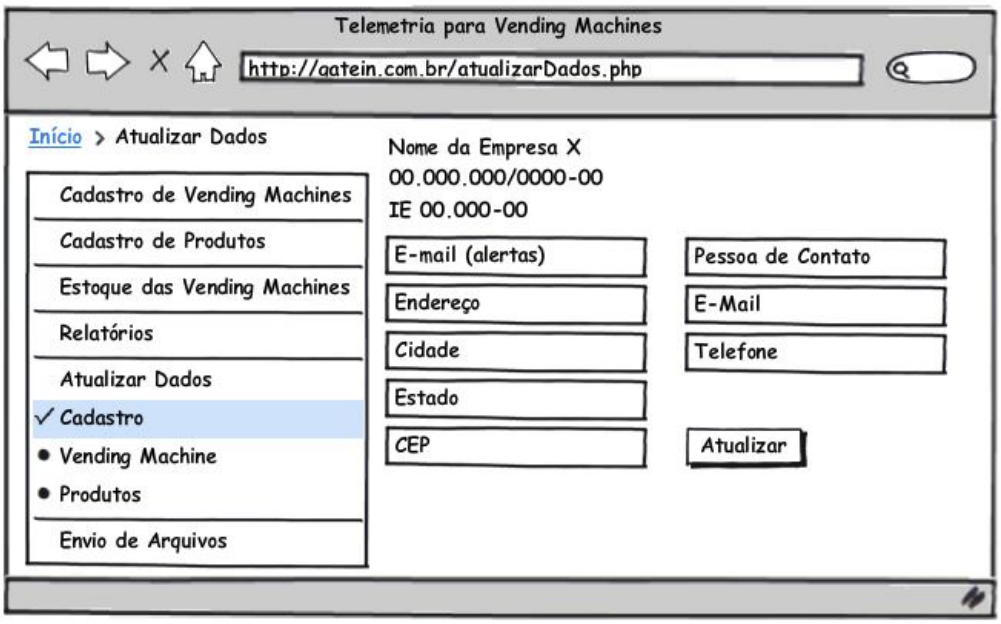


Figura 17: Esboço - Tela de Atualização de Informações

2.8. Troca de Arquivos

Arquivo: Arquivos.php

1. Permite o envio manual de arquivos para:
    a. Registro de produtos no banco de dados;
    b. Registro de *Vending Machines* e configuração de conteúdo;
2. Descreve funcionamento e protocolos para troca de arquivos XML (integração com outros sistemas)

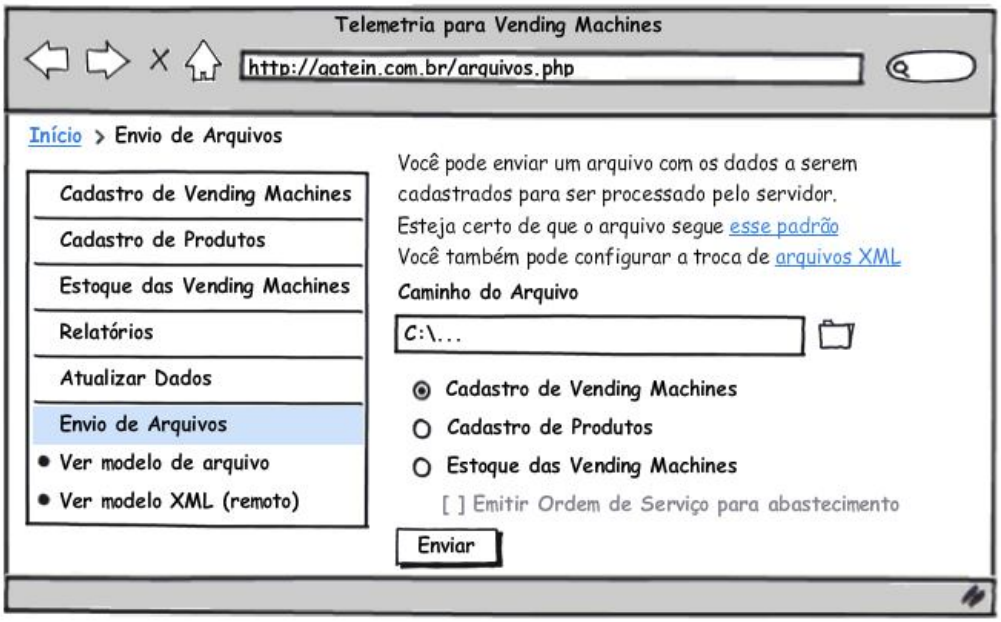

Telemetria para Vending Machines
http://gatein.com.br/arquivos.php
Início > Envio de Arquivos
Cadastro de Vending Machines
Cadastro de Produtos
Estoque das Vending Machines
Relatórios
Atualizar Dados
Envio de Arquivos
Ver modelo de arquivo
Ver modelo XML (remoto)
Você pode enviar um arquivo com os dados a serem cadastrados para ser processado pelo servidor.
Esteja certo de que o arquivo segue esse padrão
Você também pode configurar a troca de arquivos XML
Caminho do Arquivo
C:\...
Cadastro de Vending Machines
Cadastro de Produtos
Estoque das Vending Machines
[ ] Emitir Ordem de Serviço para abastecimento
Enviar

Figura 18: Esboço - Tela de Envio de Arquivos

Com base nos requisitos levantados e na organização apresentada, modelou-se o Banco de Dados de acordo com os relacionamentos demonstrados na figura a seguir.

Para que a imagem seja completamente inteligível, os campos em Azul são estruturas de dados, enquanto os campos em Vermelho são chaves referenciadas de outras tabelas. Quando o campo estiver vazado significa que o campo pode ser nulo, ou seja, o valor/relacionamento pode existir ou não.

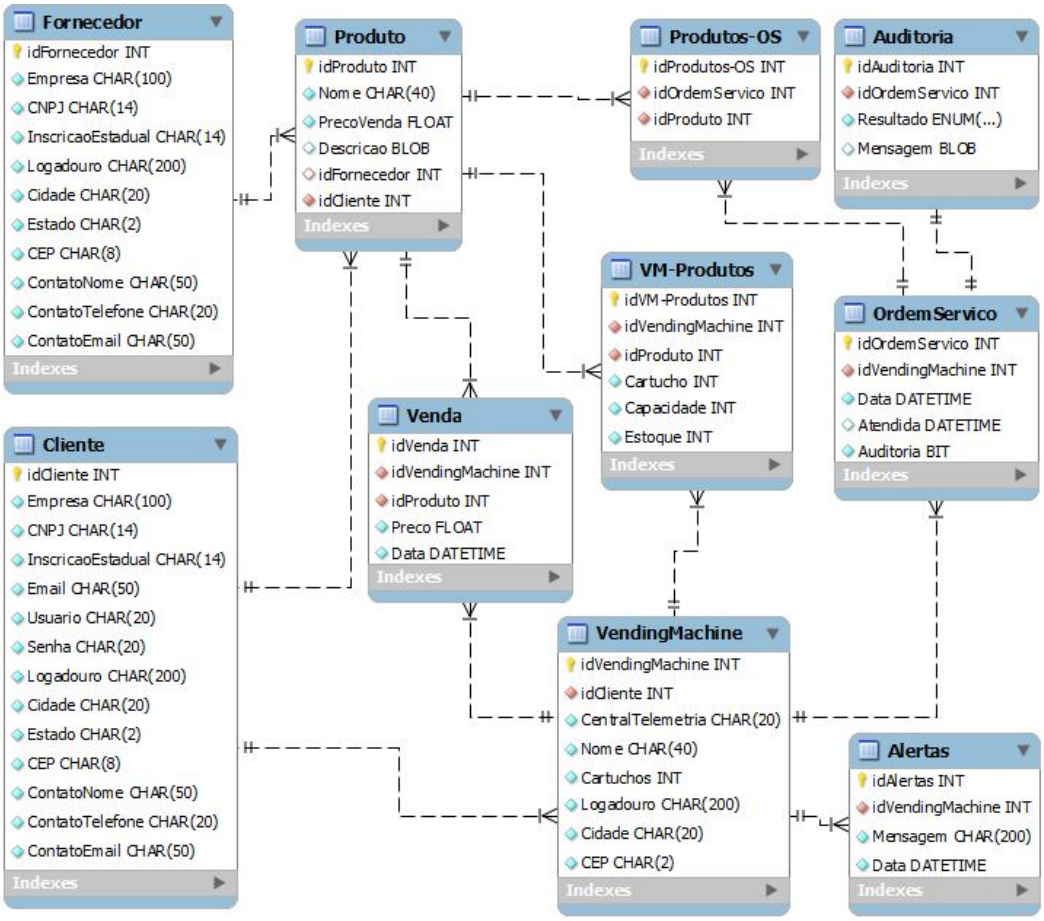


Figura 19: Diagrama Entidade-Relacionamento Completo

3. Emulador MDB

Como não tivemos acesso a uma *Vending Machine* para executar nossos testes, desenvolvemos um emulador do protocolo MDB. A função deste é enviar através de uma porta serial, um byte com 9 bits, tal qual o próprio MDB.

3.1. Funcionamento do Emulador

O emulador deve enviar um byte comum, com 8 bits, seguido de um bit indicando o tipo do dispositivo que enviou este byte: Mestre (1) ou Escravo (0). Para isso, o emulador ajusta a paridade do byte entre os dois valores possíveis, *MARK* (Marca), representando o nível lógico 1, ou *SPACE* (Espaço), com o nível lógico zero.

3.2. Usando o Emulador

Na Figura 20 está uma imagem do emulador desenvolvido.

Para enviar os dados, primeiro selecione uma porta *COM* livre do computador e clique em Conectar.

Em seguida, ajustar os bits que serão enviados, clicando nos botões dos bits. O bit MB pode ser ajustado, quando desejado. À medida que os valores vão se alterando, a representação em Decimal e Ascii vão sendo atualizadas.

Clicando em Enviar, a sequência é então enviada para a porta serial.

O botão Fechar encerra o programa.

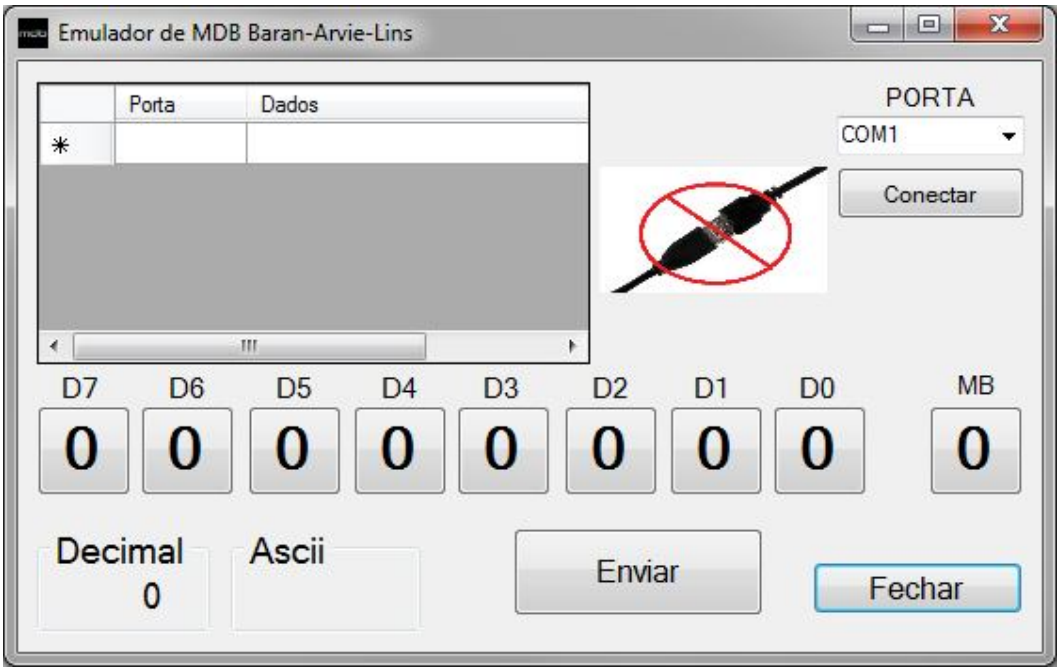


Figura 20 - Janela do Emulador MDB

Tecnologias utilizadas

1. Transmissão de Dados

Tendo em vista que o sistema é em tempo real, a melhor escolha foi sem dúvida de trabalhar com a Internet, devido às diversas consultas por parte das centrais de telemetria, e cadastramento de informações no banco de dados.

2. Sobre o Hardware

O modem usado foi o **MotorolaG24-J**. Este oferece suporte a comandos AT, baseados em comunicação serial (exemplo: *hyperterminal*). Também oferece a possibilidade de ser programado em Java, o que torna o sistema mais simples. Ele conta com um microprocessador e memória embarcados, que desempenham o papel do microcontrolador externo.

Este módulo possui um total de 16 portas de IO, sendo oito para uso geral, e as outras oito multiplexadas com outras funções do G24. A Tabela 1 mostra um resumo dos pinos disponíveis.

Tabela 1 - Pinos de GPIO

Número do pino	Direção	Multiplexado	Reservado	Interrupção
28	I/O			Sim
30	I/O			
32	I/O			
34	I/O			
36	I/O			
38	I/O			
40	I/O			
42	I/O		Sim	
41	O	Ant_det		
49	O	GPRS		
26	O	Wkupo_n		
23	I/O	R_I		
13	I/O	DSR_N		
19	I/O	DTR_N		Sim
17	I/O	DCD_N		Sim
16	I/O	Wkupi_n		Sim

Os pinos sombreados em verde são multiplexados com a *UART1*.

O Pino 42, sombreado em amarelo, é reservado pelo módulo G24, para comutar este entre o modo *Jtool Manager* – O modo de gerência do G24-J –, e o *MIDlet Manager*, de gerência e execução da MIDlet. Este pino não está disponível no projeto. A *MIDlet* é o programa em Java que roda no G24-J.

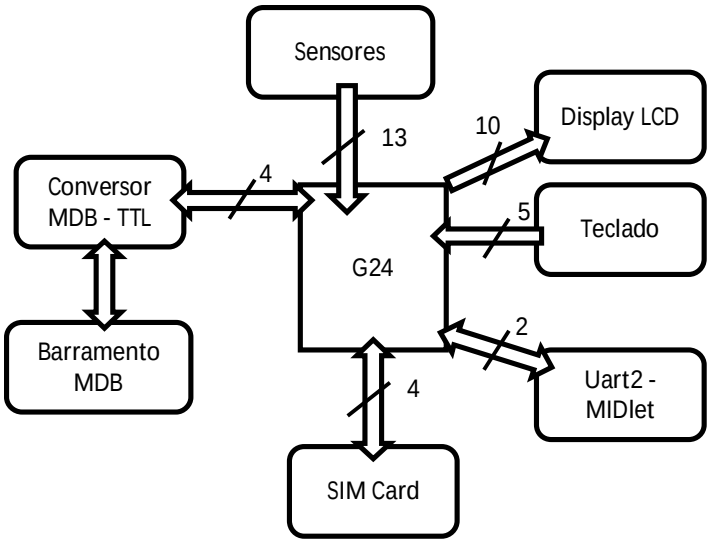


Figura 21 - Diagrama em Blocos

De acordo com o diagrama em blocos apresentado na Figura 21, para conectar todos os periféricos foram usados37 pinos.

Como o G24-J possui pinos dedicados para a conexão com o SIM Card, e com a *UART2* para programação, então restam as conexões com o barramento MDB, com o teclado, display LCD e sensores. O barramento MDB é interpretado via porta serial, pela *UART1*.As demais conexões, que foram feitas através das GPIOs, tem um total de 10 pinos de saída e 18 para entrada, sendo estes últimos 5 do teclado, e 13 dos sensores. (ou seja, 28 pinos das GPIOs).

Como este número de pinos não está disponível, foram utilizados dois modelos de *shift-registers*, um com entrada paralela e saída serial, e o outro com entrada serial e saída paralela.

Com dois *shift-registers* para entrada, em cascata, e com um para a saída, podemos ler 16 portas e escrever 8 portas, sendo que cada chip possui 8 portas.

Foram aplicados os seguintes modelos de *shift-registers*:

- CD4094, *shift-register SIPO* (*Serial Input – Parallel Output*)
- CD4021, *shift-register PISO* (*Parallel Input – Serial Output*)

Abaixo apresentamos o diagrama da montagem de dos dois circuitos em cascata, para que possam fornecer até 16 entradas, paralelamente.

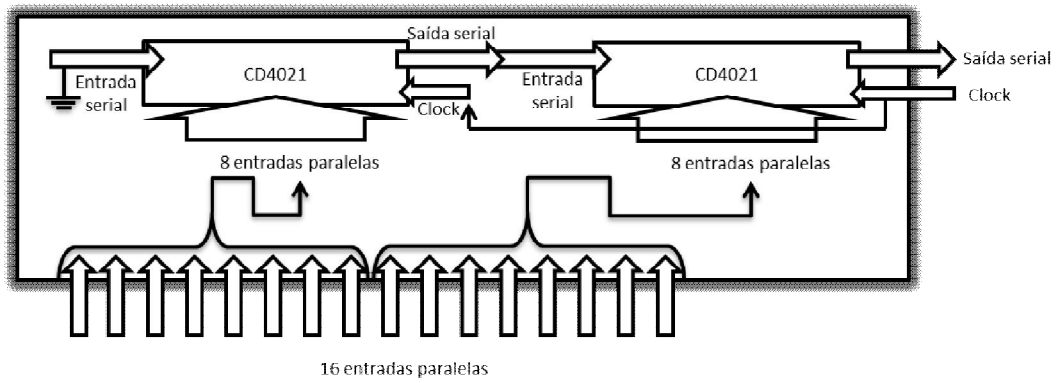


Figura 22- Blocos internos do *shift-register* para ampliação das I/Os (entrada).

Abaixo apresentamos o diagrama da montagem do circuito de saída, que fornece até 8 saídas, paralelamente.

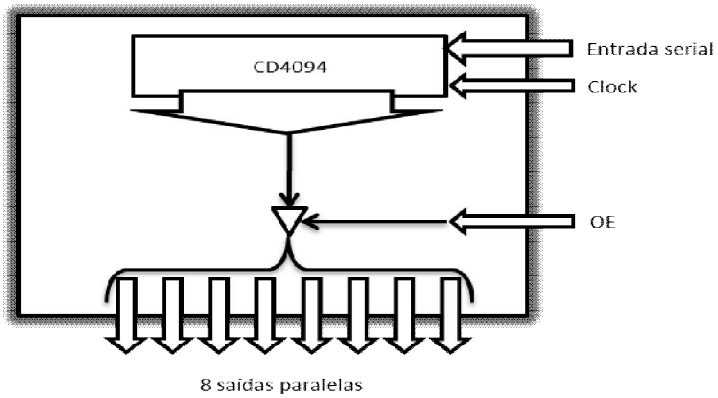


Figura 23 - Diagrama em blocos interno do *shift-register* para ampliação das I/Os (saída).

Sendo que dispomos de 16 portas de entrada, e 8 de saída, as conexões foram feitas de acordo com a seguinte tabela:

Entrada		Saída	
Nome do pino	Número da porta	Nome do pino	Número da porta
Teclado	12.15	Display LCD	0..8
Sensores	0..11		

Note-se que temos espaço para 12 sensores. Para a saída dos dados, o G24-J deve enviar os dados serialmente para o pino GPIO2, e gerar o *clock* para o *shift-register* SIPO CD4094 em GPIO3. Quando a transmissão termina, é setado o GPIO4, (*output enable*). Assim, os dados ficam disponíveis nas portas do módulo de telemetria.

Foi desenvolvido também um identificador de mudanças no barramento. Caso algum dos pinos mude de valor, o circuito gera um pulso em um pino de interrupção do G24-J (GPIO16 para sensores ou GPIO14 para teclado), informando este que uma leitura do barramento deve ser efetuada.

A cada interrupção, o G24-J entende que algo mudou no barramento de entrada. Assim, ele inicia uma leitura do barramento completo, gerando sinais de *clock*, na porta GPIO5 para o *shift-register PISO* CD4021, e lendo as respostas deste em GPIO6. Para cada pulso de *clock*, um bit consecutivo do barramento está disponível na porta de saída serial do CI.

Lendo estes bits, e os armazenando nas posições corretas de um vetor em memória do G24-J, pudemos processar os barramentos separadamente. Por exemplo, a posição 0 do vetor indica uma tecla.

Na Figura 21 apresentamos o diagrama teórico de conexão com o G24-J. Abaixo, trazemos uma representação mais real de como as conexões foram feitas, usando o modelo para expandir as portas de I/O apresentado.

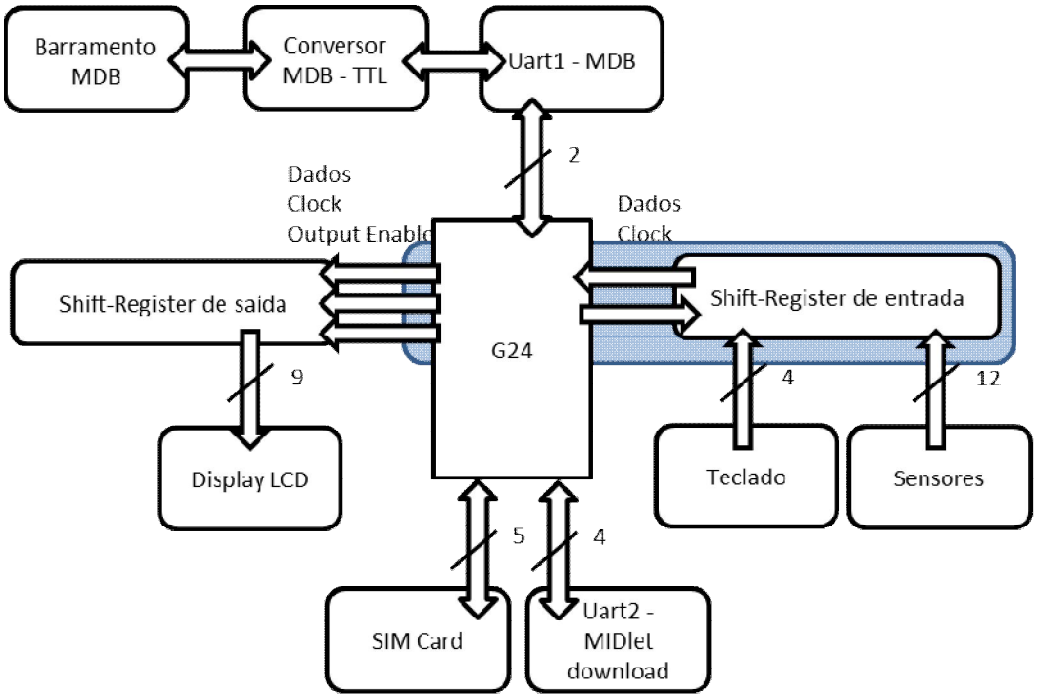


Figura 24 - Diagrama de blocos do G24 com os periféricos, usando os *shift-registers*

Podemos notar um retângulo abaixo do G24 e dos *shift-registers*. Este representa o gerenciador de interrupções comentado anteriormente.

Na figura abaixo temos o diagrama completo das conexões. Alguns dos módulos não foram utilizados neste projeto, como os de áudio ou de USB, por exemplo.

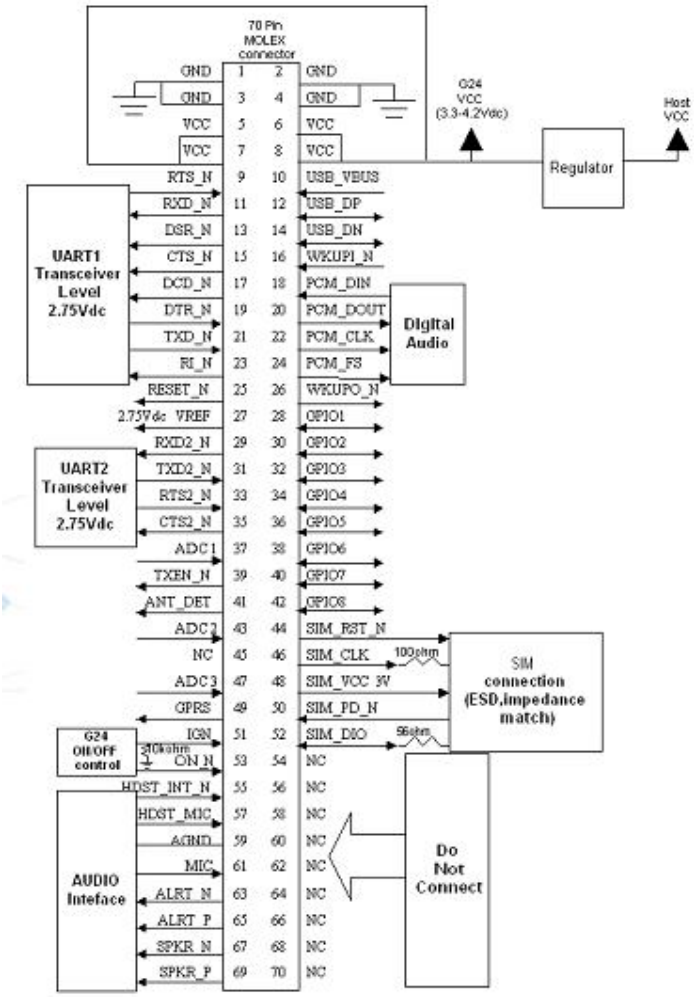


Figura 25 - Diagrama geral de conexões

Antes de ser feita a escolha deste modem para o módulo de telemetria, foi considerada a possibilidade de serem utilizados os protocolos de comunicação sem fio *Zigbee* e *Simplicity*, além da comunicação via *Ethernet*.

Em conversa com o professor Afonso Miguel, orientador deste projeto, foi descartado o uso da porta *Ethernet* já que algumas empresas não permitem que terceiros se conectem à sua rede local.

O *Zigbee* e o *Simplicity* não são protocolos de longa distância e exigiriam o desenvolvimento de um gateway *GPRS* onde pudessem se conectar para transmitir os dados ao servidor. Considerando-se a complexidade em lidar com um protocolo adicional, foi optado por instalar um modem *GPRS* diretamente em cada módulo de telemetria. Deste modo elimina-se a necessidade de se preocupar com a distribuição de repetidores de sinal, e aproveita-se a existência da rede de telefonia móvel já instalada.

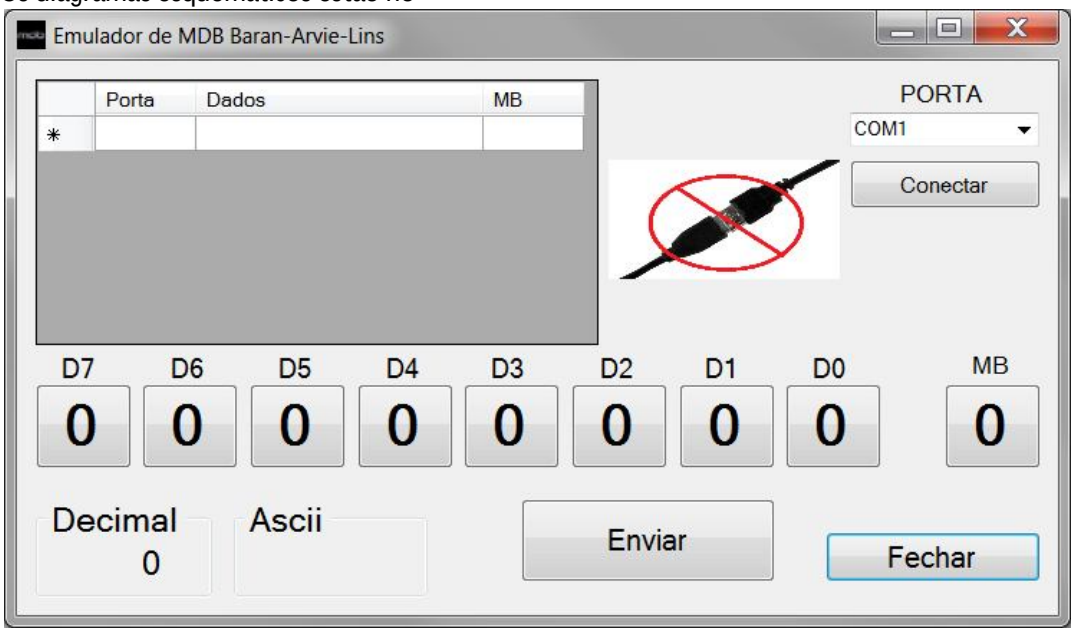
Custo:

- U$91Motorola G24-J
- R$10 Sim Card Vivo
- R$13 Franquia de 8MB de dados (Preço unitário para pacote de 200 licenças)

O hardware proposto inicialmente não sofreu grandes modificações. As diferenças mais notáveis estão na divisão da placa, inicialmente construída em apenas um módulo, para cinco módulos:

- Placa principal, ver Figura 26.
- Placa gerenciadora de Entradas, ver Figura 27**Erro! Fonte de referência não encontrada.**.
- Placa gerenciadora de Saídas e Botões de *Menu*, Figura 28.
- Suporte do G24,Figura 29.
- Conversor MDB,
- Figura 30.

Os diagramas esquemáticos estão no



Anexo 4: Layouts e Esquemáticos das placas:.

O principal motivo para a divisão em múltiplas placas foi permitir que caso houvesse problemas em uma parte do projeto, apenas esta fosse descartada. Por exemplo, na placa do suporte para o G24, com o soquete de 70 pinos, as trilhas são muito finas e muito próximas. Descobrimos que é relativamente fácil que uma delas se rompa durante a fabricação, ou que entrem em curto. Assim, podemos descartar apenas a placa na qual aconteceu o erro, como de fato aconteceu com a placa suporte para o G24, sem alterar o resto do protótipo.

As placas inicialmente seriam fabricadas por uma empresa especializada, nós apenas desenvolveríamos o *layout*. Mas, como pequenas alterações no projeto demandam uma placa inteiramente nova, achamos que não seria interessante manter esse modelo, pois o tempo que a empresa demanda para fabricar uma placa, até a sua entrega, nos causaria atrasos.

Por exemplo, segundo o site www.pci.eletronica.org/faq/, as placas demoram até seis semanas, e custam entre R$1,20 e R$1,40 por cm².

No total do nosso projeto, usamos 204,5 cm², o que levaria a um custo de R$245,70 para fabricá-las. Este é o preço para um conjunto das 5 placas, necessárias ao sistema.Este custo é apenas para as placas, sendo que os componentes podem ser reaproveitados.

Comprando um kit para desenvolver as placas em casa, usando o processo fotográfico para todas elas, o custo final seria ainda menor conforme apresentado: R$75,00no kit, mais aproximadamente R$40,00 com as placas virgens, num total de R$115,00. Uma economia de R$130 em relação à compra da placa pronta.

Foi decidido então que faríamos nossas placas usando o processo fotográfico, adquirindo um kit de fabricação de placas de circuito impresso, disponíveis no mercado.

Também havia sido proposto pintar as placas com verniz azul, mas este se mostrou muito complicado, de difícil obtenção e longa secagem. Também foi abandonado, e em seu lugar foi usado verniz normal transparente.

Como já havíamos previsto no início do projeto, e conforme foi dito anteriormente, precisaríamos de várias placas, para contornar erros de projeto. Então, apenas para efeito de ilustração, somando todas as placas que usamos durante o desenvolvimento, chegamos a 629,5 cm² de placa, o que acarretaria num custo total de R$755,40 se o serviço fosse realizado por terceiros.Esse valor ultrapassa todos os demais custos do projeto somados, além do orçamento que tínhamos estimado em investir. Com o kit de desenvolvimento evitamos gastos adicionais de R$620,40.

2.1.Funções de cada Placa

- Placa Principal:

  Contêm as conexões para todas as outras placas do sistema embarcado.

  Possui duas portas DB-9, uma para conexão serial com o emulador de MDB (Porta 1), e a outra, para auditoria, por onde as informações sobre o estoque são enviadas,

complementando as informações enviadas ao LCD (Porta 2). Esta também é a porta pela qual a programação do módulo G24 é feita, e na maior parte do tempo, estará desconectada.

Também conta com um suporte para o SIM CARD, uma fonte estabilizada para 5v e 3.8v, um conversor TTL-RS232, um botão e dois *jumpers*, para controle do G24, e alguns *leds*, para indicar o estado do sistema.

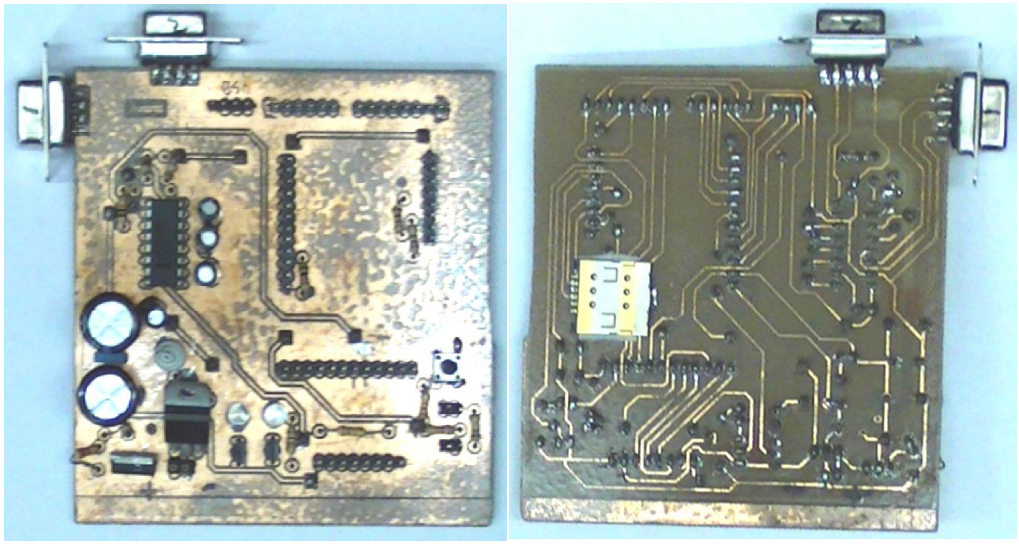


Figura 26 - Placa principal (à esquerda, face superior)

- Placa gerenciadora de Entradas:

  Monitora constantemente as entradas, compostas pelos sensores e pelos botões, localizados na placa do LCD. Caso algum deles mude de nível lógico, ela gera uma interrupção para o G24.

Contêm dois *shift-registers PISO* (*Parallel in – Serial Out*), e amplificadores operacionais para monitorar o estado dos pinos.

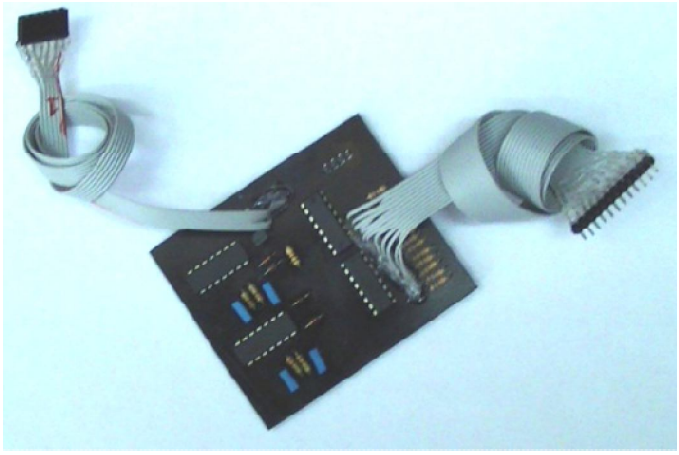


Figura 27 - Entradas, conexão com os sensores.

- Placa gerenciadora de Saídas:

  Um *shift-register SIPO (Serial in – Parallel Out)*, coloca os dados vindos do G24 na entrada do LCD.

  As interrupções para os botões são geradas aqui, separadamente das geradas pelos sensores. Há uma conexão entre as placas de entrada e saída, pois os dados (qual botão foi pressionado) são enviados pela placa de entrada. A Interrupção, indicando que qualquer botão foi pressionado, é gerada nesta placa.

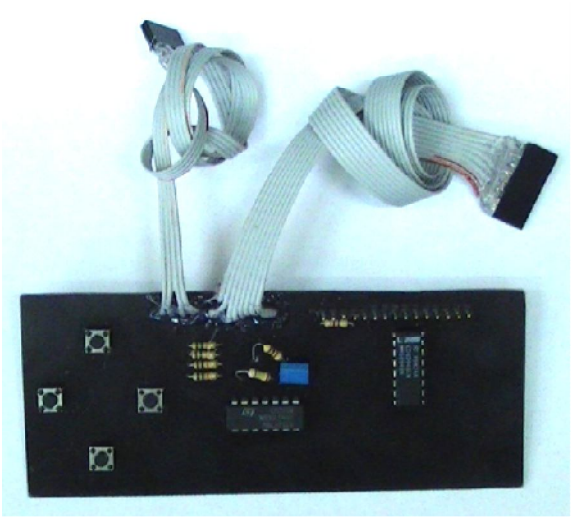


Figura 28 - Placa de Saídas e Botões.

- Suporte do G24:

  Apenas uma placa de expansão dos pinos do G24. Foi criada pelos motivos citados acima, e basicamente só faz a conexão das portas do G24 úteis ao projeto à placa principal. Também liga alguns pinos ao nível lógico baixo (0v), para que estes não tenham que ser conectados pela placa mãe, simplificando o *layout* de ambas.

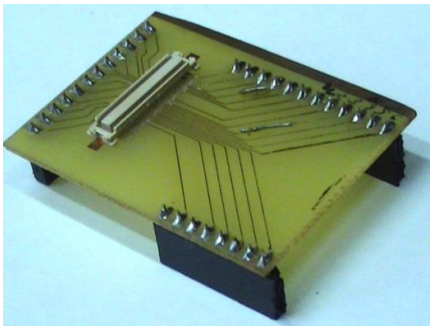


Figura 29- Suporte para o G24

- Conversor MDB:

Monitora o barramento MDB, através do micro controlador PIC. A placa também possui um cristal, para gerar o *clock* do PIC, e um *led*, para monitorar o nono bit (*Master/Slave*), transmitido pelo MDB.

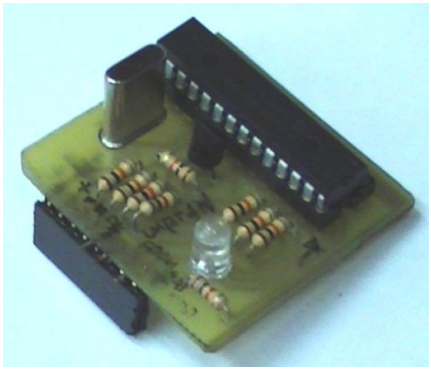


Figura 30 - Conversor MDB

1.1. Conectando as placas

O SIM CARD é conectado na placa principal (Figura 31). Nesta placa também vão ligados o conversor MDB (Figura 32), a placa do LCD (Figura 33) e a placa responsável pelos sensores (Figura 34). modem vai ligado ao suporte, que por sua vez também fica encaixado na placa. A antena vai ao conector (Figura 36) e a outra ponta do conector, ao Modem G24 (Figura 37).

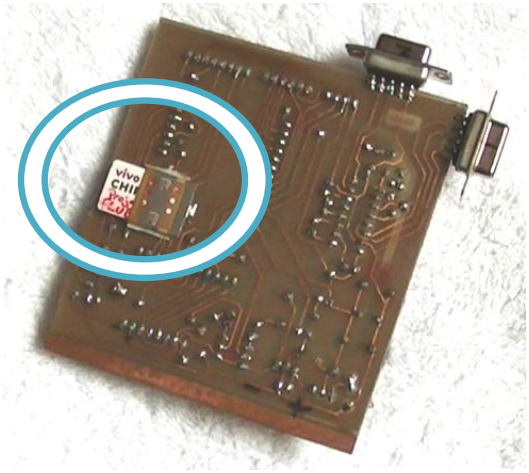


Figura 31 - Com SIM CARD.

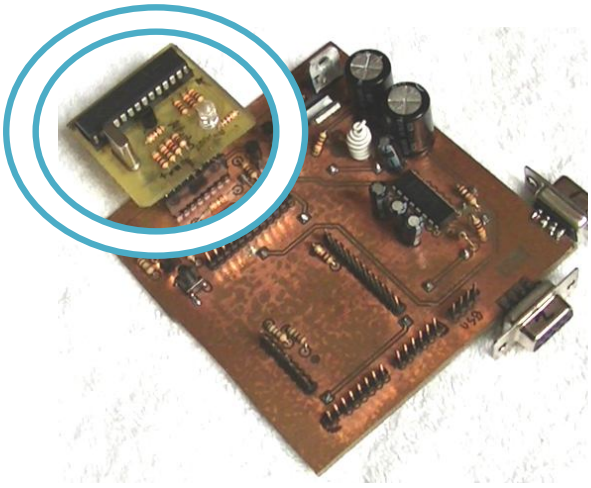


Figura 32 - Conectando o módulo conversor de MDB.

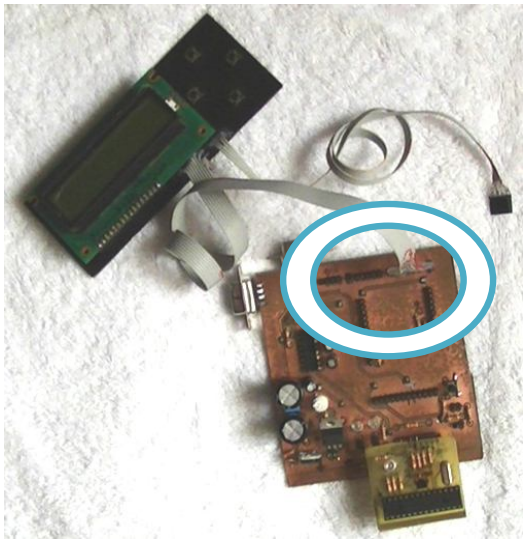


Figura 33 - Conectando a placa do LCD

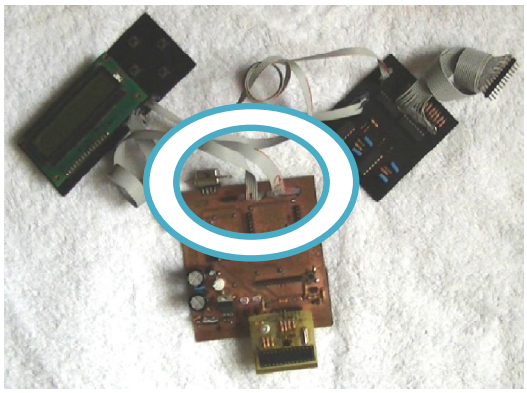


Figura 34 - Conectando a placa dos sensores.

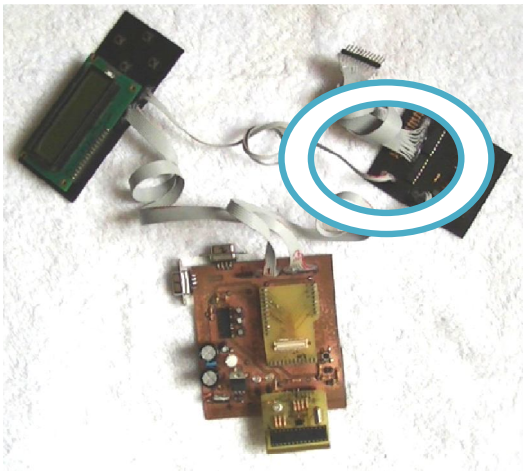


Figura 35 - Interconexão entre a placa e suporte ao Módulo G24

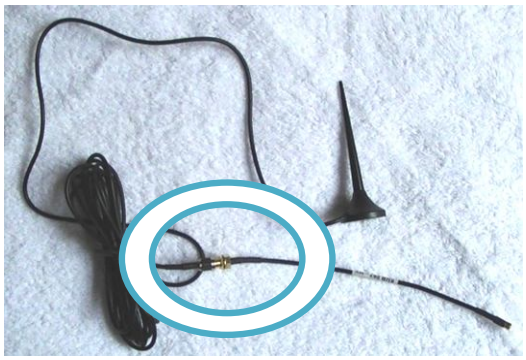


Figura 36 - Conectando o adaptador à antena.

Figura 37 - Modem conectado à antena.

2. Sobre o Firmware

Todo o firmware que roda no módulo G24 é orientado a interrupções. Isto significa que em grande parte do tempo o módulo está desocupado. Também há uma rotina de inicialização, executada junto ao servidor, quando a máquina conectada pela primeira vez (não é necessário em casos de falta de alimentação da rede elétrica.).

Todas as funções são, direta ou indiretamente, chamadas pela função:

`onInterrupt(int, boolean)` – arquivo UserMIDlet.java.

O código java que está implementado no modem encontra-se no Anexo 1: Firmware do Modem G24-JAVA.

2.1. Rotina de Inicialização

Assim que se instala uma nova *Vending Machine*, é necessário inicializá-la. A rotina de inicialização tem como objetivo fornecer ao sistema de telemetria, as informações necessárias sobre a máquina que está sendo monitorada, sobre em que cartucho está qual produto.

Ao ligar a central, acessando a função de inicialização, esta pede que se pressione os botões dos produtos carregados na máquina. Fazendo isso, a rede MDB da máquina vai gerar o código referente à entrega de cada um destes produtos. A central fica monitorando o barramento MDB, e armazena a informação presente em uma estrutura de dados da memória interna.

Como a central recebe do servidor a informação de quantos tipos diferentes de produtos esta máquina receberá, um loop é executado para que todos os cartuchos presentes na máquina sejam armazenados.

O exemplo do arquivo recebido pela central de telemetria está apresentado em *1.2 - Inicializar,* na página nº 11.

As funções responsáveis pela rotina de inicialização são:

- `Inicializa_MDB()`– arquivo MIDlet.java;
- `le_da_uart()`- arquivo Uart.java

2.2. Navegação no Menu

A navegação no *menu* é disparada por uma interrupção na linha GPIO 14. Esta dispara o processo de leitura das portas de entrada através dos *shift-registers*, já comentado. Os valores lidos do *shift-register* são então armazenados em um vetor na memória. Dependendo do botão pressionado, uma função diferente para a navegação é disparada.

As funções são:

- `back()`– arquivo Menu.java

- `para_baixo()` – arquivo Menu.java
- `para_cima()` – arquivo Menu.java
- `botao_menu()` – arquivo Menu.java

A árvore interna do *menu* é a seguinte:

```
0 = raiz
|-->1 = Status
| |-->11 = Verificar Status
| |-->12 = inicializar
| └-->12 = estoque
|-->2 = Ordens de Serviço
| |-->21 = Escolher os
| | └-->211 = Estado especial 1.
| |-->22 = Detalhar os escolhida
| | └-->221 = Estado especial 2.
| |-->23 = Executar os escolhida
| └-->24 = Completar
└-->3 = Sensores
```

```
Estado especial 1: Para baixo: os--. Para cima: os++. Back: estado =
21
Estado especial 2: Para baixo: produto--. Para cima: produto++. Back:
estado = 21
```

Para auxiliar a navegação do *menu*, outras funções foram desenvolvidas, e armazenadas em outro arquivo: Auxiliares.java.

2.3. Conexão com a Internet

Todos os serviços e processos que são executados neste sistema, estão de uma forma ou de outra, ligados à internet. Para isto, uma única função foi desenvolvida. Esta trata todas as conexões de entrada e saída com a rede.

- `static String net(String parametro, String ip, String url)` – arquivo ServiçosHttp.java

Os parâmetros desta função são:

a. parametro: Parâmetro transmitido no corpo da mensagem POST. Normalmente carrega o identificador da Central de Telemetria, mas em alguns casos pode conter também um texto formatado em XML.
b. ip: Indica o IP do servidor. Como este possui IP dinâmico, o mesmo deve ser requisitado periodicamente. No caso da presente aplicação, este é requisitado cada vez que se entra no *menu*.
c. url: É o caminho, dentro do servidor, que deve ser chamado para execução de uma determinada função.

O retorno desta função é uma String, contendo a resposta do servidor, formatada como XML. A informação desejada é extraída por funções específicas, dependendo da chamada que foi executada (para cada função, um XML diferente é retornado). São elas:

- `tira_xml_ip` – arquivo Auxiliares.java
- `tira_xml_estoque` – arquivo Auxiliares.java
- `tira_xml_status` – arquivo Auxiliares.java
- `tira_xml_inicializar` – arquivo Auxiliares.java

O retorno destas funções também é uma String, mas esta contém apenas os dados necessários.

2.4. Vendas

Quando uma mensagem circula no barramento MDB, o G24 lê esta mensagem, e a decodifica. Se esta for igual a um dos endereços de cartuchos, que foram armazenadas durante a rotina de inicialização, então uma venda é detectada. O sistema avisa o servidor, para que este tome as medidas necessárias. A função usada para a venda é:

- `trata_MDB-` arquivo UserMIDlet.java.

O fluxo de dados e os modelos das mensagens trocadas foram apresentados em 1.2 - Inicializar, na p. 11.

2.5. Alertas e Sensores

O sistema fica constantemente monitorando o estado das 12 entradas para sensores. Quando uma delas muda de nível lógico, um alerta é enviado ao servidor pela função apresentada abaixo:

- *`Trata_Sensores`* - arquivo UserMIDlet.java

Responsável por monitorá-los, e disparar os alertas.

O fluxo de dados e os modelos das mensagens trocadas foram apresentados em 1.6 - Alertas, na página nº 16.

2.6. Ordens de Serviço

As ordens de serviço são solicitadas do servidor pela função OS do *menu*.

- `pega_oss` – arquivo Auxiliares.java

Responsável por solicitar ao servidor os dados das Ordens de Serviço. Esta função preenche algumas estruturas de dados com as informações sobre as ordens de serviço disponíveis para esta máquina. Elas ficam então disponíveis para serem visualizadas e operadas utilizando as opções do *menu*. É possível selecionar uma Ordem de Serviço, detalhar, e executá-la.

O fluxo de dados e os modelos das mensagens trocadas foram apresentados em *1.4 - Ordens de Serviço*, na página nº 12.

Para a navegação dentre as ordens, e dos produtos de cada ordem, funções de apoio foram criadas:

- `pega_os_anterior` – arquivo Auxiliares.java
- `pega_proxima_os` – arquivo Auxiliares.java
- `pega_produto_anterior` – arquivo Auxiliares.java
- `pega_proximo_produto` – arquivo Auxiliares.java

2.7. Escrita no LCD

As funções a seguir atuam sobre o LCD, com capacidade de escrever Strings completas, enviar comandos como apagar e saltar para a linha inferior ou superior:

- `escreve_sem_pular_linha(String s)` - arquivo LCD_Mgr.java
  - Escreve no máximo os primeiros 16 caracteres da String s.
- `escreve(String s)` - arquivo LCD_Mgr.java
  - Escreve os primeiros 16 caracteres da String s. Se ela contiver mais que 16 caracteres, a função quebra a linha e continua a escrita até atingir o limite de 32 caracteres.
- `pula_linha(String string)` - arquivo LCD_Mgr.java
  - Salta para a linha “1” ou “0”, passada como parâmetro.
- `LCD_clear_and_reset-` arquivo LCD_Mgr.java
- `inicia_lcd-` arquivo LCD_Mgr.java

2.8. Conversor 9bits(MDB) -> 8bits

Esta parte do firmware não é roda no G24, mas em um PIC16F876A, situado na placa do conversor (Ver

Figura 30).

O barramento é monitorado em uma porta sensível a mudança de nível. Quando acontece tal mudança, o sistema transfere os primeiros 8 bits, que correspondem ao byte de dados, para a Uart1 do G24. O último bit é colocado em uma porta GPIO do G24, e uma interrupção é então gerada neste. Assim, o firmware do G24 lê o dado da Uart1 e da porta definida, e pode proceder com sua execução.

O firmware do conversor está no

Anexo 2: Firmware do Conversor 9bits (MDB) -> 8bits.

3. Barramento MDB (Multi-Drop Bus)

O meio de comunicação utilizado entre os periféricos e a VMC ou entre a VMC e os periféricos é a comunicação serial com *Baud Rate* de 9600. A comunicação funciona da seguinte forma:

Serial Bit Format:1 Start Bit + 8 Data Bits + 1 Mode Bit + 1 Stop Bit = 11 Bits Total

(LSB) Start, 8 Data Bits, Mode, Stop (MSB)

Em uma transmissão de uma VMC para um periférico, o bit Mode diferencia se a transmissão trata-se de uma transmissão de endereço ou de dados. No caso da transmissão tratar-se de endereço, o bit Mode estará em nível lógico um e todos os periféricos irão ler a transmissão. Quando a transmissão for de dados, o bit Mode estará em nível lógico zero, e será somente aceita pelo periférico que foi endereçado, na operação anterior.

No caso de uma transmissão de um periférico para o VMC, o bit *Mode* deverá ser setado no ultimo byte de dados enviado.

Em uma comunicação entre VMC e periféricos, os bits 7..3 equivalem aos bits de endereço. Um bloco de dados é definido pelo endereço, dados opcionais e CHK(checksum). O limite de tamanho é de 36 bytes. A resposta de uma VMC a um periférico é um Acknowledgment (ACK), Negative Acknowledgment (NAK), or Retransmit (RET). Os códigos dos mesmos são descritos abaixo.

- ACK 00H (acknowledgment/checksum correto)
- RET AAH (Retransmita a informação anterior. Esta mensagem só pode ser enviada pela VMC)
- NAK FFH (Negativo acknowledge)

A VMC e o periférico classificam uma espera de 5ms após o inicio da comunicação como uma mensagem NAK.

Existem especificações de endereço para periféricos que são:

Address Definition

00000xxxB (00H) Reserved for VMC
00001xxxB (08H) Changer
00010xxxB (10H) Cashless Device #1
00011xxxB (18H) Communications Gateway
00100xxxB (20H) Display
00101xxxB (28H) Energy Management System
00110xxxB (30H) Bill Validator
00111xxxB (38H) Reserved for Future Standard Peripheral

01000xxxB (40H) Universal Satellite Device #1
01001xxxB (48H) Universal Satellite Device #2
01010xxxB (50H) Universal Satellite Device #3
01011xxxB (58H) Coin Hopper or Tube – Dispenser 1
01100xxxB (60H) Cashless Device #2
01101xxxB (68H) Coin Hopper or Tube – Dispenser 2
01110xxxB (70H) Reserved for Future Standard Peripherals
. . .
11011xxxB (D8H) Reserved for Future Standard Peripherals
11100xxxB (E0H) Experimental Peripheral #1
11101xxxB (E8H) Experimental Peripheral #2
11110xxxB (F0H) *Vending Machine* Specific Peripheral #1
11111xxxB (F8H) *Vending Machine* Specific Peripheral #2

Além das especificações de endereço, também foram especificadas funções padrões para alguns tipos de periféricos.

Procedimentos de Teste e Validação Durante o Desenvolvimento

1. Modem GPRS

Inicialmente apresentava-se como o maior desafio do projeto por se tratar de uma tecnologia nova para os envolvidos. Foram feitos testes de troca de mensagens básicas (“Olá mundo!”) até que alcançada a estabilidade no sistema de comunicação.

Os módulos foram capazes de enviar e receber mensagens HTTP, interpretar estas mensagens e tomar as atitudes corretas dependendo do conteúdo da mensagem.

Testes em caixa preta:

1 - Uma mensagem deve ser enviada do módulo para um servidor escolhido, e este deve a reconhecer.

- Caso de sucesso: O servidor interpreta corretamente a mensagem.
- Caso de fracasso: O servidor não consegue receber a mensagem.
    - Como resolver:
    - Verificar a conexão com a internet
    - Verificar a presença do SIM Card, com créditos para tráfego GPRS
    - Verificar a disponibilidade do servidor
- Caso de fracasso: O servidor recebe, mas não consegue interpretar a mensagem.
    - Resetar o modem GPRS

2 – Uma mensagem deve ser recebida pelo modem G24-J, em resposta à mensagem de teste anterior, acendendo um LED indicando sucesso da recepção, e outro indicando sucesso na interpretação a mensagem.

- Caso de sucesso: O modemG24-J recebe e interpreta corretamente a mensagem.
- Caso de fracasso: O modemG24-J não consegue receber a mensagem.
    - Verificar a conexão com a internet
    - Verificar a presença do SIM Card, com créditos para tráfego GPRS
- Caso de fracasso: O modem G24-J recebe, mas não consegue interpretar a mensagem.
    - Resetar o modem G24-J

Testes em caixa branca:

Não existem para o modem G24-J isoladamente, uma vez que não podemos acessar seu sistema embarcado.

2. Módulo de Telemetria

Depois de alcançado o controle sobre o recebimento de mensagens, foi feita a instalação de um modem no módulo de telemetria. Primeiramente foram feitos testes sobre este módulo através de LED's permitindo identificar as operações que estavam sendo realizadas. Sinais foram injetados diretamente nos pinos de interesse.

Operações incorretas não são aceitas. O sistema é capaz de se recuperar sozinho no caso de falha de comunicação, ou então gerar uma mensagem de erro para o administrador do sistema e/ou das VM's.

Quando as operações do módulo foram realizadas adequadamente, foi introduzida a comunicação GPRS. Quando for verificado um funcionamento pleno do módulo em parceria com o servidor, foi dado início ao processo de desenvolvimento de um protótipo de interface. Este dispositivo teve como simples funcionalidade simular operações de venda, reabastecimento, etc., sem a necessidade de um software de debug ou aplicação direta de sinais sobre o módulo.

Testes em caixa preta:

1 - Uma mensagem deve ser enviada do módulo de telemetria, indicando uma venda, e o servidor deve receber esta notificação.

Caso de sucesso: O servidor interpreta corretamente a mensagem.

- Caso de fracasso: O servidor não consegue receber a mensagem.
    - Como resolver:
    - Verificar a conexão com a internet.

    - Verificar a presença do SIM Card, com créditos para tráfego GPRS.
    - Verificar as conexões no barramento MDB.
    - Verificar a disponibilidade do servidor.
- Caso de fracasso: O servidor recebe, mas não consegue interpretar a mensagem.
    - Resetar o modem GPRS.
    - Verificar conexões no barramento MDB.

2 - Uma mensagem deve ser enviada do módulo de telemetria para o servidor, indicando uma falha (mensagem de erro ou de sensor) e este deve a reconhecer.

- Caso de sucesso: O servidor interpreta corretamente a mensagem.
- Caso de fracasso: O servidor não consegue receber a mensagem.
    - Verificar a conexão com a internet.
    - Verificar a presença do SIM Card, com créditos para tráfego GPRS.
    - Verificar conexões com os sensores.
    - Verificar a disponibilidade do servidor.
- Caso de fracasso: O servidor recebe, mas não consegue interpretar a mensagem.
    - Resetar o modem GPRS.
    - Verificar conexões com os sensores.

3 – Um dos botões do módulo é pressionado e seu respectivo número é exibido no LCD do módulo.

- Caso de sucesso: O G24-J responde a interrupção e manda as informações ao display.
- Caso de fracasso: O número do botão não é exibido corretamente.
    - Verificar dados exibidos
    - Verificar conexões com a placa do LCD
- Caso de fracasso: Nada é exibido no LCD
    - Verificar alimentação do módulo LCD

4 – Uma das chaves representando os sensores é ligada ou desligada, e seu estado é exibido no LCD.

- Caso de sucesso: O G24-J responde a interrupção e manda as informações ao display.
- Caso de fracasso: O número do sensor, ou seu valo não é exibido corretamente.
    - Verificar dados exibidos
    - Verificar conexões com a placa de entrada dos sensores
- Caso de fracasso: Nada é exibido no LCD
    - Verificar alimentação do módulo LCD
    - Verificar conexões com o módulo LCD
    - Verificar alimentação do módulo de entrada dos sensores.

<u>Testes em caixa branca:</u>

Os testes são os mesmos, mudando o procedimento de correção, nos casos de fracasso.

1 - Uma mensagem deve ser enviada do módulo de telemetria, indicando uma venda, e o servidor deve receber esta notificação.

Caso de sucesso: O servidor interpreta corretamente a mensagem.

- Caso de fracasso: O servidor não consegue receber a mensagem.
    - Verificar a programação do módulo, buscando por erros na conexão com a internet ou na interpretação do MDB.
    - Verificar a placa do módulo, buscando erros de soldagem, ou componentes queimados ou danificados.
    - Verificar a fonte de alimentação, assegurando que o modem está corretamente alimentado.
- Caso de fracasso: O servidor recebe, mas não consegue interpretar a mensagem.
    - Verificar a programação do módulo, buscando por erros na interpretação do MDB.

2 - Uma mensagem deve ser enviada do módulo de telemetria para o servidor, indicando uma falha (mensagem de erro ou de sensor) e este deve a reconhecer.

- Caso de sucesso: O servidor interpreta corretamente a mensagem.
- Caso de fracasso: O servidor não consegue receber a mensagem.
    - Verificar a programação do módulo, buscando por erros na conexão com a internet ou na interpretação do MDB.
    - Verificar a placa do módulo, buscando erros de soldagem, ou componentes queimados ou danificados.
    - Verificar a fonte de alimentação, assegurando que o modem está corretamente alimentado.
    - Verificar o circuito gerenciador de interrupções, garantindo que está gerando as interrupções para cada alteração no barramento paralelo dos sensores.

3 – Um dos botões do módulo é pressionado e seu respectivo número é exibido no LCD do módulo.

- Caso de sucesso: O G24-J responde a interrupção e manda as informações ao display.
- Caso de fracasso: O número do botão não é exibido corretamente.
    - Verificar soldas na placa do LCD
    - Verificar conexões com a placa do LCD
- Caso de fracasso: Nada é exibido no LCD
    - Verificar alimentação do módulo LCD
    - Verificar *backlight* do LCD, bem como seu contraste.

4 – Uma das chaves representando os sensores é ligada ou desligada, e seu estado é exibido no LCD.

- Caso de sucesso: O G24-J responde a interrupção e manda as informações ao display.
- Caso de fracasso: O número do sensor, ou seu valo não é exibido corretamente.
    - Verificar resistores de *pull-down* no módulo de entrada.
    - Verificar conexões com a placa de entrada dos sensores
- Caso de fracasso: Nada é exibido no LCD
    - Verificar alimentação do módulo LCD
    - Verificar conexões com o módulo LCD, *backlight* e contraste
    - Verificar alimentação do módulo de entrada dos sensores.

Procedimentos de Teste e Validação Para a Defesa do Projeto

Como estes procedimentos serão executados durante a apresentação, a maioria das soluções para os possíveis erros consistem em apenas a re-execução do método, pois não é possível alterar códigos do servidor durante a apresentação, por exemplo.

1. Procedimentos Executados exclusivamente no Servidor

1. O servidor deve permitir o cadastro de um novo usuário no sistema.
   - Caso de Sucesso: Cadastro efetuado e novo usuário funcional
   - Caso de fracasso: Servidor não aceita cadastro.

2. Cadastro de Fornecedores.
   - Caso de Sucesso: Fornecedor registrado no sistema.
   - Caso de fracasso: Servidor não aceita cadastro.

3. Cadastro de Produtos
   - Caso de Sucesso: Produto disponível no sistema.
   - Caso de fracasso: Servidor não aceita cadastro.

4. Cadastro de *Vending Machine*s
   - Caso de Sucesso: *Vending Machine* registrada no sistema, disponível para registrar o conteúdo de seus cartuchos.
   - Caso de fracasso: Servidor não aceita cadastro.

5. Registro de Conteúdo na *Vending Machine*
   - Caso de Sucesso: Produtos da *Vending Machine* ficam registrados no sistema. A máquina pode ser inicializada, através de uma Ordem de Serviço.
   - Caso de fracasso: A máquina fica sem o estoque definido, ou um estoque diferente do desejado fica registrado.

6. Geração de Ordens de Serviço
   - Caso de Sucesso: Ordem de serviço registrada no sistema, e disponível para a central de telemetria.
   - Caso de fracasso: A Ordem de Serviço não fica cadastrada no sistema, ou produtos incorretos são cadastrados na Ordem de Serviço.

7. Alteração de Estoque
   - Caso de Sucesso: O sistema permite a alteração do estoque de uma determinada *Vending Machine*.

- Caso de fracasso: A alteração do estoque não acontece, ou acontece de forma diferente da esperada.

8. Alteração de Cadastro
   - Caso de Sucesso: Os dados do cadastro do usuário são alterados no cadastro.
   - Caso de fracasso: A alteração não acontece, ou dados incorretos são armazenados.

9. Alteração de Produtos
   - Caso de Sucesso: Os dados do produto são alterados.
   - Caso de fracasso: Os dados do produto não são alterados corretamente.

10. Consulta do Histórico de Vendas
    - Caso de Sucesso: O histórico das vendas foi mantido corretamente, e é exibido de acordo.
    - Caso de fracasso: Não é exibida nenhuma venda, ou dados incorretos são exibidos.

11. Consulta do Histórico de Alertas
    - Caso de Sucesso: O histórico de alertas foi mantido corretamente, e é exibido de acordo.
    - Caso de fracasso: Não é exibido nenhum alerta, ou dados incorretos são exibidos.

2. Procedimentos executados exclusivamente na *Vending Machine*

1. Inicialização da central – capturando os dados do Barramento MDB
   - Caso de Sucesso: Os dados são corretamente armazenados na memória do G24.
   - Caso de fracasso: Os dados não são armazenados ou são armazenados incorretamente.

2. Consulta do estado dos sensores
   - Caso de Sucesso: Os estados dos sensores são exibidos no LCD corretamente.
   - Caso de fracasso: Os estados dos sensores não são exibidos ou exibidos incorretamente.

3. Procedimentos executados em conjunto – VM e Servidor

1. Atendimento de Ordens de Serviço
   - Caso de Sucesso: As ordens são atendidas e o servidor processa os dados destas ordens.
   - Caso de fracasso: As ordens de serviço não são atendidas ou o servidor não consegue processar corretamente os dados recebidos.

2. Vendas
    - Caso de Sucesso: A máquina detecta uma venda no barramento MDB e envia um XML para o servidor, que é recebido e processado com sucesso.
    - Caso de fracasso: A máquina não detecta a venda.
    - Caso de fracasso: O servidor não recebe a venda.
    - Caso de fracasso: O servidor não processa a venda.

3. Gerando Alertas
    - Caso de Sucesso: A máquina identifica uma mudança de estado nos sensores e gera um XML de alerta para o servidor, que recebe e processa corretamente.
    - Caso de fracasso: A máquina não identifica uma mudança de estado nos sensores.
    - Caso de fracasso: O servidor não processa o alerta da máquina.

Análise de Riscos

Os riscos no projeto foram:

ID	Descrição	Pro	Imp	Sev	Ação/Prevenção	Contingência
1	Dificuldade na integração do módulo GPRS com o servidor e/ou módulo de telemetria.	Média	Alto	Alta	Consulta a documentação e especialistas	Substituição do modem GPRS por porta Ethernet para comunicação via Internet.
2	Baixa disponibilidade de sinal para rede de telefonia móvel.	Baixa	Médio	Média	Reposicionamento do módulo dentro da VM e/ou colocação de antena externa.	Reposicionamento da VM.
3	Atraso na entrega dos módulos por causa de motivos de saúde.*	Baixa	Alto	Alta	Realinhamento de cronograma com prazos previstos e recalculados.	Utilização de versão similar dos módulos.
4	Atraso na entrega dos módulos por quaisquer motivos que impeçam o andamento normal do cronograma.	Baixa	Alto	Alta	Realinhamento de cronograma com prazos previstos e recalculados.	Utilização de versão similar dos módulos.
5	Impedimento na programação dos módulos G24/G24-j por dificuldades na transmissão do programa para o	Média	Alto	Alta	Consulta a manuais específicos, apoios técnicos da Motorola, e especialistas.	Substituição do modem G24-J pelo modelo G24, sem suporte a Java, com um microcontrolador

	modem					auxiliar (PIC/8051), ou por outro modelo de comunicação
6	Falha na comunicação via WEB, do modem para o servidor, por quaisquer erros de programação do modem.	Média	Alto	Alta	Testes da programação do modem, através de simulações e testes de conexão simples (“olá mundo”).	Recodificação do programa no modem, outras medidas cabíveis.
7	Erro durante a soldagem do conector do módulo G24-J	Alta	Médio	Baixa	Realização da solda usando equipamentos especiais para soldagem de SMDs.	Aquisição de placa pronta ou solda por encomenda junto a especialistas (p.ex. técnicos de celular).
8	Falha no projeto pela falta dos componentes necessários no mercado	Baixa	Médio	Baixo	Pesquisa de mercado para verificar onde podemos adquirir tais componentes	Substituição dos componentes por similares.

Definição dos acrônimos utilizados:

- Pro – Probabilidade
- Imp – Impacto
- Sev – Severidade

*Como, por exemplo, mas não limitado ao impacto da segunda fase da H1N1.

Viabilidade e Conclusões

O objetivo do projeto resumiu-se a produzir um módulo de telemetria para *Vending Machine*s. Conceitualmente, os conhecimentos adquiridos durante o desenvolvimento do projeto podem ser aplicados em sistemas similares para praticamente qualquer tipo de serviço ou produto, o que é um aspecto de interesse para os membros da equipe.

Sob o ponto de vista comercial nota-se um custo elevado no desenvolvimento de cada módulo dado o preço do modem G24-J, que chega a ser superior ao preço de um celular básico. No entanto o custo mensal de manutenção do sistema é bastante baixo, tornando a opção uma proposta comercialmente viável para longo prazo.

O projeto foi desenvolvido para finalidades acadêmicas, agregando conhecimentos na área de telemetria e comunicação por redes GPRS. Acredita-se que tais áreas de conhecimento podem ser amplamente exploradas num futuro próximo, dada a crescente demanda por

sistemas de automação e monitoramento nas mais variadas áreas de aplicação (residencial, comercial, indústria, governo, etc.).

Consideramos que o sistema atende seus objetivos básicos ao ser capaz de notificar um servidor remoto sobre operações de venda e abastecimento, situações básicas considerando-se o funcionamento de uma *Vending Machine*.

Mesmo tendo havido uma redução no escopo do projeto, acreditamos que o resultado final atendeu aos objetivos inicialmente propostos, uma vez que um sistema de telemetria é uma ferramenta para o auxílio de tomada de decisão. Através do produto final desenvolvido pudemos trabalhar com decisões autônomas (criação de ordens de serviço mediante regras pré-definidas), envio automático de notificações ao usuário, coleta e organização de dados permitindo posterior consulta por terceiros.

Cronograma

Tarefa	Maio	Junho	Julho	Agosto	Setembro	Outubro	Novembro	Dezembro
Detalhamento do projeto físico	1	20						
Aquisição de modems GPRS e testes de comunicação com PC	1	30						
Entrega do projeto físico		21						
Defesa do projeto físico		21						
Revisão do projeto físico -- Módulo de Telemetria (hardware e software)			1		17			
Implementação do servidor, incluindo modelos de arquivos XML			1	30	Atraso 30			
Testes e validações preliminares com o servidor			15	30	Atraso 30			
Testes de comunicação do modem GPRS em circuito dedicado (para uso futuro nomódulo)			1		30			
Elaboração do manual de comunicação (troca de arquivos XML com o servidor)					1	30	Atraso 21	
Entrega do projeto físico revisado.					20			
Desenvolvimento do módulo de telemetria						1	21	
Homologação da comunicação entre central e servidor							1 21	
Elaboração do relatório final						1	14 21	
Revisão do manual de comunicação							1 21	
Elaboração do manual do módulo de telemetria							1 21	
Entrega do relatório final e manual de implementação e comunicação							22	01
Defesa da implementação final							22	01

Legenda:

Tarefa terminada
Tarefa atrasada
Tarefa prevista
Tarefa adiada
Tarefa corrente

Referências Bibliográficas

Para a elaboração deste documento, foram úteis os seguintes documentos, programas e páginas da Web:

MySQL Workbench, disponível on-line em:
http://dev.mysql.com/downloads/workbench/5.2.html [Acesso em Maio de 2010]

Analizador de arquivos DTS, disponível on-line em:
http://evadts.dejongduke.nl/[Acesso em Maio de 2010]

*Vending Machine*s, disponível on-line em:
http://www.bonusdata.net/vending/AuditData/vending.htm/[Acesso em Maio de 2010]

Hardware description do modem Motorola G24, disponível on-line em:
http://developer.motorola.com/docstools/developerguides/G24_HardwareDescription_Developer_Guide.pdf/[Acesso em Maio de 2010]

Motorola KJava (Manual de programação em Java para o modem Motorola G24-J), disponível on-line em:
http://www.motorola.com/staticfiles/Business/Products/M2M%20Wireless%20Modules/G24%20Lite/_Documents/static%20files/G24%20KJava_New.pdf?localeId=33[Acesso em Maio de 2010]

PCI.eletrônica (Website por onde se podem encomendar Placas de Circuito Impresso). Disponível on-line em:
www.pci.eletronica.org/faq/
[Acesso em Julho de 2010]

Anexos

Anexo 1: Firmware do Modem G24-JAVA

1. *Arquivo UserMIDlet.java:*

```
package g24;

import java.io.IOException;
import java.util.Random;

import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

import com.motorola.oem.hapi.GpioException;
import com.motorola.oem.hapi.GpioInput;
import com.motorola.oem.hapi.GpioInterruptConfig;
import com.motorola.oem.hapi.GpioInterruptListener;
import com.motorola.oem.osc.OSC;
import com.motorola.oem.osc.OSCException;

public class UserMIDlet extends MIDlet implements
GpioInterruptListener  {

	public static boolean[] entrada = new boolean[16];
	public static boolean[] entrada_old = new boolean[16];
	private GpioInterruptConfig gpioInterruptConfig;
	int cont = 0, selecionado = 0, indice = 1;
	static int index = -1;
	Random randomGenerator = new Random();
	private static GpioInput MasterSlave, InteruptMDB;


	/**
	 *
	 * Estados do menu:
	 * 0 = raiz
	 * 1 = Status
	 * 11 = inicializar
	 * 12 = estoque
	 * 2 = Ordens de Servico
	 * 21 = seleciona os corrente
	 * 211 = Detalhes da os corrente
	 * 212 = Executar a os corrente
	 * 22 = Completar
	 * 3 = Sensores
	 *
	 * */

	public UserMIDlet() {
		OSC.stopMIDletWatchdog();
		System.out.println("My First G24 MIDlet.");
	}

	protected void destroyApp(boolean unconditional)
			throws MIDletStateChangeException {
	}
```

```
	protected void pauseApp() {
	}

	//Avantt Hypertechnologies
	protected void startApp() throws MIDletStateChangeException {

		gpioInterruptConfig = new GpioInterruptConfig(this);

		try {
			Uart.inicia_uart1();
			LCD_Mgr.inicia_lcd();
			ServicosGpio.inicia(gpioInterruptConfig);
			LCD_Mgr.escreve("-Inicializando-");
			ServicosHttp.conectar();
			ServicosGpio.wkupo.write(true);

			MasterSlave = new GpioInput(12);
			InteruptMDB = new GpioInput(15);

			InteruptMDB.enableInterrupt(gpioInterruptConfig);

			Menu.ip = Auxiliares.pega_ip();

	//ServicosHttp.net("CentralTelemetria="+Menu.nome_central,Menu.i
p,"clear");

			LCD_Mgr.LCD_clear_and_reset();
			LCD_Mgr.escreve("Pronto!");

			ServicosGpio.le_tudo(entrada);

		} catch (GpioException e) {

			System.out.println("Erros!");
			e.printStackTrace();
		} catch (Exception e) {
			System.out.println("Erros no LCD!!!!");
			System.out.println(e);
			e.printStackTrace();
		}

		System.out.println("End Start App (really!)");
	}

	 private static void showDateTime(byte []dateTime) {

	   int year = dateTime[0];
	   if (year < 2000) {
	       year += 2000;
	   }
	   String time = "";

	   if (dateTime[3] < 10) {
	       time += "0";
	   }

	   time += dateTime[3];
	   time += ":";

	   if (dateTime[4] < 10) {
```

```
            time += "0";
         }
         time += dateTime[4];
         time += ":";

          if (dateTime[5] < 10) {
            time += "0";
         }
         time += dateTime[5];

         System.out.println("Date & time: " + dateTime[2] + " in " +
dateTime[1] + " " + year + "  " + time);
 }

      public void onInterrupt(int gpioNum, boolean val){

            //System.out.println("GPIO " + gpioNum + " is " + val);


            byte b;

            if (OSC.isDateTimeAvailable())
            {
                  System.out.println("Date & time is available");
                  try {
                      byte[] dateTime=OSC.getDateTime();
                      showDateTime(dateTime);
                  } catch (OSCException osce) {
                      osce.printStackTrace();
                  }
              }

            //
            //----------------------------------------
            //

            if((!val)&&(gpioNum == 16))//Sensores
            {
                  Trata_Sensores();
            }

            //
            //----------------------------------------
            //

            if((!val)&&(gpioNum == 15))//Interrupção MDB
            {
                  if(Menu.inicializa)
                  {
                        b = Uart.le_da_uart()[0];

      ((produto)Menu.produtos_desta_maquina.elementAt(index)).MDBcode
= b;
                        Inicializa_MDB();
                  }

                  else//Operação normal
                  {
                        try {
                              System.out.println("MasterSlave = " +
MasterSlave.read());
```

```
                        } catch (GpioException e) {

                              e.printStackTrace();
                        }
                        b = Uart.le_da_uart()[0];
                  trata_MDB(b);
            }
      }

      //
      //---------------------------------------
      //

      if((!val)&&(gpioNum == 14))//Teclados
      {
            if(Menu.inicializa)
            {
                  //Ignorar teclas. Pedir que pressione o botão
na máquina, para pegar, via MDB o comando
                  //que indica o produto.
                  if(index == -1)//primeira interrupção
                        Inicializa_MDB();
                  else
                  {
                        System.out.println("Vamos incializar
alguns registros na máquina. ");
                        System.out.println("Por favor, pressione
o botão referente ao produto indicado. ");
                  }
            }

            else
            try {
                  ServicosGpio.le_tudo(entrada);

                  if(entrada[12])
                  {
                        System.out.println("Botão Menu");
                        Menu.botao_menu();
                  }

                  if(entrada[13])
                  {
                        System.out.println("Botão para cima ");
                        Menu.para_cima();
                  }

                  if(entrada[14])
                  {
                        System.out.println("Botão para baixo");
                        Menu.para_baixo();
                  }

                  if(entrada[15])
                  {
                        System.out.println("Botão back");
                        Menu.back();
                  }
```

```
                  /*
                   // Rotina para Emulador:
                      try{
                  if(gpioNum == 1)//Teste
                        {
                              Auxiliares.pega_ip();
                              Auxiliares.inicializar();
                              ServicosHttp.produtos_desta_maquina();

                              //
      System.out.println(Menu.produtos_desta_maquina.elementAt(0));
                        }

                        if(gpioNum == 14)
                        {
                              produto p;
                              //Simulando que chegou da Uart a venda do
produto 1 - Azera.

                              ServicosHttp.produtos_desta_maquina();
                              p =
pega_produto_pelo_cartucho(1);//numero do cartucho
                              venda(p); //Testando as vendas

                              System.out.println("Botão Menu");
                              Menu.botao_menu();
                        }

                        if(gpioNum == 16)//if botão para cima
                        {
                              System.out.println("Botão para cima " );
                              Menu.para_cima();
                        }

                        if(gpioNum == 15)//if botão para baixo
                        {
                              System.out.println("Botão para baixo");
                              Menu.para_baixo();
                        }

                        if(gpioNum == 1)//if(back)
                        {
                              System.out.println("Botão back" );
                              Menu.back();
                        }*/

                  } catch (Exception e1) {

                        e1.printStackTrace();
                  }
            }

      }//fim da on_interrupt

      private static void Trata_Sensores() {
            try {
                  ServicosGpio.le_tudo(entrada);
                  System.out.println("\n---------Sensores:-----------
");
                  for(int i = 0; i < 16; i++)
```

```
                        System.out.println("entrada[" + i + "] = " +
entrada[i]);
                  } catch (GpioException e) {
                  e.printStackTrace();
                  }

                  System.out.println("----------------------------
\n");

                  String alerta = "<Conteudo><Alertas>";
                  boolean alertar = false;

                  for(int i = 0; i < 12; i++)
                  {
                        if(entrada[i] != entrada_old[i])
                        {
                              alerta += "<Alerta><Mensagem>S" + i +" =
" + entrada[i] +"</Mensagem><Sensor>" + i + "</Sensor><Estado>"+
entrada[i] +"</Estado><DataHora>1988-06-03
24:00:00</DataHora></Alerta>";
                              alertar = true;
                              entrada_old[i] = entrada[i];
                        }
                  }
                  if(alertar)
                  {
                        alerta += "</Alertas></Conteudo>";
                        try {

      ServicosHttp.net("CentralTelemetria="+Menu.nome_central +
"&XML=" + alerta, Menu.ip, "alerta");
                        } catch (IOException e) {
                              e.printStackTrace();
                        }
                  }
      }

      private static int Inicializa_MDB() {

            produto p;
            if(index == -1)
            {
                  System.out.println("Vamos incializar alguns registros
na máquina. ");
                  System.out.println("Por favor, pressione o botão
referente ao produto indicado. ");
                  try {
                        LCD_Mgr.LCD_clear_and_reset();
                        LCD_Mgr.escreve_sem_pular_linha("Pressione
botao");

                        ServicosHttp.produtos_desta_maquina();
                        index = Menu.produtos_desta_maquina.size();

                        LCD_Mgr.pula_linha("1");

                        index--;

                        System.out.println("Index:" + index);
                        if(index <= 0)
                        {
```

```
                              Menu.inicializa = false;//Terminou a
rotina de inicialização
                              return 0;
                        }

                        try {
                              p =
(produto)Menu.produtos_desta_maquina.elementAt(index);
                              LCD_Mgr.escreve_sem_pular_linha("" +
p.id_produto);
                              System.out.println("Id: " +
p.id_produto);

                              } catch (GpioException e) {
                                    e.printStackTrace();
                              }

                  }
                catch (Exception e) {
                  e.printStackTrace();
                }
              }
              else
              {
                  if(index == 0)
                  {
                        Menu.inicializa = false;
                        try {
                              Menu.back();
                        } catch (Exception e) {
                              e.printStackTrace();
                        }
                  }

                  index--;

                  try {
                        p =
(produto)Menu.produtos_desta_maquina.elementAt(index);
                        LCD_Mgr.pula_linha("1");
                        LCD_Mgr.escreve_sem_pular_linha("" +
p.id_produto);
                        System.out.println("Agora, pressione o botão do
produto: " + p.id_produto);
                        } catch (GpioException e) {
                              e.printStackTrace();
                        }
              }

              return index;

      }

      private void trata_MDB(byte b) {

              produto a;

              for(int i = 0; i < Menu.produtos_desta_maquina.size();
i++)//para todos os produtos desta maquina
              {
```

```
                  if(b == ((produto)
Menu.produtos_desta_maquina.elementAt(i)).MDBcode)
                  {
                        a = (produto)
Menu.produtos_desta_maquina.elementAt(i);
                        System.out.println("Venda do produto " +
a.id_produto);
                        venda(a);
                        break;
                  }
            }

      }


      /*private produto pega_produto_pelo_cartucho(int cartucho) {

            produto p = null;
            for(int i = 0; i < Menu.produtos_desta_maquina.size(); i
++)
            {
                  p = (produto)
Menu.produtos_desta_maquina.elementAt(i);
                  if(p.cartucho == cartucho)
                        break;
            }
            return p;
      }*/

      /*
      <Vendas>
             <Produto>
                                 <idXML>Código Qualquer</idXML>
                           <idProduto>___</idProduto>
                           <Cartucho>___</Cartucho>
                           <Preco>___</Preco>
                           <DataHora>ANO/MÊS/DIA hh:mm:ss</DataHora>
             </Produto>
             */

      private void venda(produto p2) {
            String s = "<Vendas><Produto>";
            indice = randomGenerator.nextInt(100);
            s += "<idXML>" + indice + "</idXML>" + "<idProduto>"+
p2.id_produto +"</idProduto>" +
                        "<Cartucho>"+ p2.cartucho
+"</Cartucho><Preco>"+p2.preco+"</Preco>";


            s += "<DataHora>1988/06/03 12:00:01</DataHora>";

            s += "</Produto></Vendas>";

            try {

      ServicosHttp.net("CentralTelemetria="+Menu.nome_central +
"&XML=" + s, Menu.ip, "venda");
            } catch (IOException e) {
                  e.printStackTrace();
            }
```

```
	}
}
```

2. Arquivo Menu.java:

```
package g24;

import java.util.Vector;

import com.motorola.oem.hapi.GpioException;

public class Menu {

	public static int estado_atual = 0, os_id = 0, vetor_indice = 0;
	static String ip;
	static String nome_central = "grapevine";
	static String os;

	public static boolean[] entrada = new boolean[16];
	static Vector nome_de_produto = new
Vector(),quantidade_de_produto = new Vector(), ID_os = new Vector(),
data_os = new Vector();
	public static Vector produtos_desta_maquina = new Vector();
	public static boolean inicializa = false;
	public static Vector produtos_estoque = new Vector();
	public static int prod_estoque_indice = 0;

	static public void back() throws Exception {
		switch (estado_atual)
		{
		case 1:
		case 2:
		case 3:
			System.out.println("--- Raiz ---");
			LCD_Mgr.LCD_clear_and_reset();
			LCD_Mgr.escreve_sem_pular_linha("Raiz");
			estado_atual = 0;
			break;

		case 11:
		case 12:
		case 13:
			System.out.println("--- MENU ---");
			System.out.println(">>>1 = Status");
			System.out.println("2 = OS");
			System.out.println("3 = Sensores");
			LCD_Mgr.LCD_clear_and_reset();
			LCD_Mgr.escreve_sem_pular_linha("1 = Status");
			estado_atual = 1;
			break;

		case 21:
		case 22:
		case 23:
		case 24:

			System.out.println("--- MENU ---");
			System.out.println("1 = Status");
			System.out.println(">>>2 = OS");
			System.out.println("3 = Sensores");
```

```
            LCD_Mgr.LCD_clear_and_reset();
            LCD_Mgr.escreve_sem_pular_linha("2 = OS");
            estado_atual = 2;
            break;

      case 131:
            System.out.println("11 = Verificar status");
            System.out.println("12 = Inicializar");
            System.out.println(">>>13 = Estoque");
            LCD_Mgr.LCD_clear_and_reset();
            LCD_Mgr.escreve_sem_pular_linha("13.Estoque");
            estado_atual = 13;
            break;

      case 211:
      case 221:
            System.out.println(">>>21 = Escolher os");
            System.out.println("22 = Detalhar os escolhida");
            System.out.println("23 = Executar os escolhida");
            System.out.println("24 = Completar");
            LCD_Mgr.LCD_clear_and_reset();
            LCD_Mgr.escreve_sem_pular_linha("21.Escolher OS");
            estado_atual = 21;
            break;

      }
}

static public void para_baixo() throws Exception {

      switch (estado_atual)
      {
      case 1:
            System.out.println("--- MENU ---");
            System.out.println("1 = Status");
            System.out.println(">>>2 = OS");
            System.out.println("3 = Sensores");
            LCD_Mgr.LCD_clear_and_reset();
            LCD_Mgr.escreve_sem_pular_linha("2 = OS");
            estado_atual = 2;
            break;
      case 2:
            System.out.println("--- MENU ---");
            System.out.println("1 = Status");
            System.out.println("2 = OS");
            System.out.println(">>>3 = Sensores");
            LCD_Mgr.LCD_clear_and_reset();
            LCD_Mgr.escreve_sem_pular_linha("3 = Sensores");
            estado_atual = 3;
            break;

      case 11:
            System.out.println("11 = Verificar status");
            System.out.println(">>>12 = Inicializar");
            System.out.println("13 = Estoque");
            LCD_Mgr.LCD_clear_and_reset();
            LCD_Mgr.escreve_sem_pular_linha("12.Inicializar");
            estado_atual = 12;
            break;
      case 12:
            System.out.println("11 = Verificar status");
```

```
                System.out.println("12 = Inicializar");
                System.out.println(">>>13 = Estoque");
                LCD_Mgr.LCD_clear_and_reset();
                LCD_Mgr.escreve_sem_pular_linha("13.Estoque");
                estado_atual = 13;
                break;
          case 21:
                System.out.println("21 = Escolher os");
                System.out.println(">>>22 = Detalhar os");
                System.out.println("23 = Executar os");
                System.out.println("24 = Completar");
                LCD_Mgr.LCD_clear_and_reset();
                LCD_Mgr.escreve_sem_pular_linha("22.Detalhar OS");
                estado_atual = 22;
                break;
          case 22:
                System.out.println("21 = Escolher os");
                System.out.println("22 = Detalhar os");
                System.out.println(">>>23 = Executar os");
                System.out.println("24 = Completar");
                LCD_Mgr.LCD_clear_and_reset();
                LCD_Mgr.escreve_sem_pular_linha("23.Executar OS");
                estado_atual = 23;
                break;
          case 23:
                System.out.println("21 = Escolher os");
                System.out.println("22 = Detalhar os");
                System.out.println("23 = Executar os");
                System.out.println(">>>24 = Completar");
                LCD_Mgr.LCD_clear_and_reset();
                LCD_Mgr.escreve_sem_pular_linha("24.Completar");
                estado_atual = 24;
                break;

          case 131: //prod_estoque --
                if(   prod_estoque_indice > 0)
                            prod_estoque_indice--;

                Auxiliares.atualiza_lcd_com_o_produto_atual((produto)
produtos_estoque.elementAt(prod_estoque_indice));
                break;

          case 211: //os --
                os_id = Auxiliares.pega_os_anterior(os_id, ID_os);

      Auxiliares.atualiza_lcd_com_o_numero_da_os_atual(os_id, ID_os);
                System.out.println("OS atual: " +
ID_os.elementAt(os_id));
                LCD_Mgr.LCD_clear_and_reset();
                LCD_Mgr.escreve_sem_pular_linha("OS n: " +
ID_os.elementAt(os_id));
                break;

          case 221://Produto --
                vetor_indice =
Auxiliares.pega_produto_anterior(ID_os, vetor_indice);
                Auxiliares.detalhes_os(nome_de_produto,
quantidade_de_produto, vetor_indice);
                break;
          }
      }
```

```
static public void para_cima() throws Exception {
	switch (estado_atual)
	{
	case 2:
		System.out.println("--- MENU ---");
		System.out.println(">>>1 = Status");
		System.out.println("2 = OS");
		System.out.println("3 = Sensores");
		LCD_Mgr.LCD_clear_and_reset();
		LCD_Mgr.escreve_sem_pular_linha("1 = Status");
		estado_atual = 1;
		break;
	case 3:
		System.out.println("--- MENU ---");
		System.out.println("1 = Status");
		System.out.println(">>>2 = OS");
		System.out.println("3 = Sensores");
		LCD_Mgr.LCD_clear_and_reset();
		LCD_Mgr.escreve_sem_pular_linha("2 = OS");
		estado_atual = 2;
		break;
	case 12:
		System.out.println(">>>11 = Verificar status");
		System.out.println("12 = Inicializar");
		System.out.println("13 = Estoque");
		LCD_Mgr.LCD_clear_and_reset();
		LCD_Mgr.escreve_sem_pular_linha("11.Verif. Status");
		estado_atual = 11;
		break;
	case 13:
		System.out.println("11 = Verificar status");
		System.out.println(">>>12 = Inicializar");
		System.out.println("13 = Estoque");
		LCD_Mgr.LCD_clear_and_reset();
		LCD_Mgr.escreve_sem_pular_linha("12.Inicializar");
		estado_atual = 12;
		break;
	case 22:
		System.out.println(">>>21 = Escolher os");
		System.out.println("22 = Detalhar os");
		System.out.println("23 = Executar os");
		System.out.println("24 = Completar");
		LCD_Mgr.LCD_clear_and_reset();
		LCD_Mgr.escreve_sem_pular_linha("21.Escolher OS");
		estado_atual = 21;
		break;
	case 23:
		System.out.println("21 = Escolher os");
		System.out.println(">>>22 = Detalhar os");
		System.out.println("23 = Executar os");
		System.out.println("24 = Completar");
		LCD_Mgr.LCD_clear_and_reset();
		LCD_Mgr.escreve_sem_pular_linha("22.Detalhar OS");
		estado_atual = 22;
		break;
	case 24:
		System.out.println("21 = Escolher os");
		System.out.println("22 = Detalhar os");
		System.out.println(">>>23 = Executar os");
		System.out.println("24 = Completar");
```

```
                  LCD_Mgr.LCD_clear_and_reset();
                  LCD_Mgr.escreve_sem_pular_linha("23.Executar OS");
                  estado_atual = 23;
                  break;

            case 131: //prod_estoque --
                  if(   prod_estoque_indice < produtos_estoque.size() -
1)
                        prod_estoque_indice++;

                  Auxiliares.atualiza_lcd_com_o_produto_atual((produto)
produtos_estoque.elementAt(prod_estoque_indice));
                  break;

            case 211: //os ++
                  os_id = Auxiliares.pega_proxima_os(os_id, ID_os);

      Auxiliares.atualiza_lcd_com_o_numero_da_os_atual(os_id, ID_os);
                  System.out.println("OS atual: " +
ID_os.elementAt(os_id));
                  break;

            case 221://Produto ++
                  vetor_indice = Auxiliares.pega_proximo_produto(ID_os,
vetor_indice);
                  Auxiliares.detalhes_os(nome_de_produto,
quantidade_de_produto, vetor_indice);
            break;
            }
      }

      static public void botao_menu() throws Exception {

            /**
             *
             * Estados do menu:
             * 0 = raiz
             * .1 = Status
             * ..11 = Verifica Status
             * ..12 = inicializar
             * ..13 = estoque
             * ...131 = Estado especial. <Para baixo: prod_estoque--.
Para cima: prod_estoque++. Back: estado = 13>
             * .2 = Ordens de Servico
             * ..21 = Escolher os
             * ...211 = Estado especial. <Menu = Escolhe OS. Para
baixo: os--. Para cima: os++. Back: estado = 21>
             * ..22 = Detalhar os escolhida
             * ...221 = Estado especial. <Menu = nao faz nada. Para
baixo: produto--. Para cima: produto++. Back: estado = 21>
             * ..23 = Executar os escolhida
             * ..24 = Completar
             * .3 = Sensores
             *
             * */
            switch (estado_atual)
            {
            case 0:
                  ip = Auxiliares.pega_ip();
                  estado_atual = 1;
                  System.out.println("--- MENU ---");
```

```
                  System.out.println(">>>1 = Status");
                  System.out.println("2 = OS");
                  System.out.println("3 = Sensores");
                  LCD_Mgr.LCD_clear_and_reset();
                  LCD_Mgr.escreve_sem_pular_linha("Menu");
                  LCD_Mgr.pula_linha("1");
                  LCD_Mgr.escreve_sem_pular_linha("1 = Status");
                  break;
            case 1:
                  estado_atual = 11;
                  System.out.println(">>>11 = Verificar status");
                  System.out.println("12 = Inicializar");
                  System.out.println("13 = Estoque");
                  LCD_Mgr.LCD_clear_and_reset();
                  LCD_Mgr.escreve_sem_pular_linha("11.Verif. Status");

                  break;
            case 11:
                  System.out.println("Verifica status");
                  LCD_Mgr.LCD_clear_and_reset();
                  LCD_Mgr.escreve_sem_pular_linha("-Verif. Status-");
                  Auxiliares.testa_estado_da_maquina();
                  break;
            case 12:
                  System.out.println("Inicializa");
                  LCD_Mgr.LCD_clear_and_reset();
                  LCD_Mgr.escreve_sem_pular_linha("-Inicializa-");
                  Auxiliares.inicializar();
                  break;
            case 13:
                  System.out.println("Estoque");
                  LCD_Mgr.LCD_clear_and_reset();
                  LCD_Mgr.escreve_sem_pular_linha("-Estoque-");
                  Auxiliares.verifica_estoque();
                  estado_atual = 131;
                  break;
            case 2:

      if(Auxiliares.pega_oss(os,nome_de_produto,quantidade_de_produto,
ID_os, data_os))
                  {
                        System.out.println(">>>21 = Escolher os");
                        System.out.println("22 = Detalhar os");
                        System.out.println("23 = Executar os");
                        System.out.println("24 = Completar");
                        LCD_Mgr.LCD_clear_and_reset();
                        LCD_Mgr.escreve_sem_pular_linha("21.Escolher
OS");
                        estado_atual = 21;
                  }
                  else
                  {
                        System.out.println("--- MENU ---");
                        System.out.println("1 = Status");
                        System.out.println(">>>2 = OS");
                        System.out.println("3 = Sensores");
                        System.out.println(" NAO TEM OSS NAO
INSISTA!");
                        LCD_Mgr.LCD_clear_and_reset();
                        LCD_Mgr.escreve_sem_pular_linha("2 = OS");
                        LCD_Mgr.pula_linha("1");
```

```
                        LCD_Mgr.escreve_sem_pular_linha("NAO TEM
OS!!");
                        estado_atual = 2;
                  }

                  break;
            case 21:
                  os_id = 0;

                  try {
                        LCD_Mgr.escreve("Escolha");
                        LCD_Mgr.pula_linha("1");
                        LCD_Mgr.escreve("OS id:" +
ID_os.elementAt(os_id));
                  } catch (GpioException e) {
                        e.printStackTrace();
                  }
                  System.out.println("Escolher OS: Aperte para baixo ou
para cima.");
                  System.out.println("OS atual: " +
ID_os.elementAt(os_id));
                  estado_atual = 211;
                  break;
            case 22:
                  System.out.println("Detalhar OS: Aperte para baixo ou
para cima para ver os produtos.");
                  LCD_Mgr.LCD_clear_and_reset();
                  LCD_Mgr.escreve_sem_pular_linha("-Detalhar OS-");
                  estado_atual = 221;
                  break;
            case 23:
                  Auxiliares.executa_os_corrente(nome_central, ip,
(String) ID_os.elementAt(os_id));
                  LCD_Mgr.LCD_clear_and_reset();
                  LCD_Mgr.escreve_sem_pular_linha("!Executa!");
                  break;
            case 24:
                  Auxiliares.completar();
                  LCD_Mgr.LCD_clear_and_reset();
                  LCD_Mgr.escreve_sem_pular_linha("!Completa!");
                  break;
            case 3:
                  Auxiliares.estado_sensores(entrada);
                  LCD_Mgr.LCD_clear_and_reset();
                  LCD_Mgr.escreve_sem_pular_linha("!Entrada!");
                  LCD_Mgr.pula_linha("1");
                  ServicosGpio.le_tudo(entrada);
                  LCD_Mgr.sensores(entrada);

                  break;

            }
      }

}//fim da classe
```

3. Arquivo Auxiliares.java

```
package g24;
```

```
import java.io.IOException;
import java.util.Vector;

import com.motorola.oem.hapi.GpioException;

public class Auxiliares
{


	public static String pega_ip()
	{
		String ip = "erro";
		try {
			ip = ServicosHttp.net("CentralTelemetria=" +
Menu.nome_central ,null, "ip");
			ip = tira_xml_ip(ip);
		} catch (IOException e)
		{
			e.printStackTrace();
		}
		return ip;
	}

	public static void atualiza_lcd_com_o_numero_da_os_atual(int
os_indice, Vector idOS) {

		try{
		LCD_Mgr.pula_linha("1");
		LCD_Mgr.escreve("" + idOS.elementAt(os_indice));
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}

	}

	public static String tira_xml_ip(String ip) { //ok!

		return ip.substring(ip.indexOf("<IP>") + 4,
ip.indexOf("</IP>"));
	}

	static void verifica_estoque()
	{
		try {

	tira_xml_estoque(ServicosHttp.net("CentralTelemetria=" +
Menu.nome_central,Menu.ip, "status"));

			} catch (IOException e) {

			e.printStackTrace();
		}
	}

	private static void tira_xml_estoque(String net) {

		produto a = null;

		//System.out.println("++++++++++net = " + net);
```

```
            try
            {
                  net =
net.substring(net.indexOf("<Cartuchos>"),net.indexOf("</Cartuchos>")-
12);
                  int inicio = 0, fim = net.lastIndexOf('<');
                  Menu.produtos_estoque.removeAllElements();

                  while((inicio < fim)&&(inicio >= 0))
                  {
                        a = new produto(

      Integer.parseInt(net.substring(net.indexOf("<idProduto>",inicio)
+ 11 , net.indexOf("</idProduto>", inicio))),

      Integer.parseInt(net.substring(net.indexOf("<Numero>", inicio) +
8, net.indexOf("</Numero>", inicio))),
                                          0,
                                          (byte) 0,

      net.substring(net.indexOf("<Nome>",inicio) + 6 ,
net.indexOf("</Nome>", inicio))
                                          );

                        a.estoque_atual =
Integer.parseInt(net.substring(net.indexOf("<Atual>",inicio) + 7 ,
net.indexOf("</Atual>", inicio)));
                        a.capacidade_max =
Integer.parseInt(net.substring(net.indexOf("<Capacidade>",inicio) + 12
, net.indexOf("</Capacidade>", inicio)));

                        System.out.println("-----cartucho: " +
a.cartucho);
                        System.out.println("id produto: " +
a.id_produto);
                        System.out.println("capacidade: " +
a.capacidade_max);
                        System.out.println("atual: " +
a.estoque_atual);
                        System.out.println("mínimo: " +
net.substring(net.indexOf("<Minimo>",inicio) + 8 ,
net.indexOf("</Minimo>", inicio)));
                        System.out.println("nome: " + a.nome);
                        System.out.println("preço: " +
net.substring(net.indexOf("<Preco>",inicio) + 7 ,
net.indexOf("</Preco>", inicio)));
                        System.out.println("descrição: " +
net.substring(net.indexOf("<Descricao>",inicio) + 11 ,
net.indexOf("</Descricao>", inicio)));
                        inicio = net.indexOf("</Cartucho>",++inicio);

                        Menu.produtos_estoque.addElement(a);
                  }

                  LCD_Mgr.LCD_clear_and_reset();
                  LCD_Mgr.escreve_sem_pular_linha("Pres. p baixo ou");
                  LCD_Mgr.pula_linha("1");
                  LCD_Mgr.escreve_sem_pular_linha("p cima");

            }
            catch(Exception e){
```

```
                  System.out.println("Estoque indisponível");
                  try
                  {
                        LCD_Mgr.LCD_clear_and_reset();
                        LCD_Mgr.escreve_sem_pular_linha("Est. nao
existe");
                  }
                  catch(Exception e2)
                  {

                  }
            }

      }

      private static String tira_xml_status(String net) { //ok!

            if(net.indexOf("<id>") > 0)//tem status
            {
                  /*if(Menu.ano < 2010)

      seta_data_hora(net.substring(net.indexOf("<Data>") + 6,
net.indexOf("<Data>")));*/

                  return net.substring(0, net.indexOf("<", 0)) +
                  " - " + net.substring(net.indexOf("<id>", 0) + 4,
net.indexOf("</id>", 0)) +
                  " - " + net.substring(net.indexOf("<mensagem>", 0) +
10, net.indexOf("</mensagem>", 0)) ;
            }
            else //n tem status.
            {
                  return net.substring(0, net.indexOf("<", 0)) +
                  " - " + net.substring(net.indexOf("<Mensagem>", 0) +
10, net.indexOf("</Mensagem>", 0)) ;
            }
      }

      public static void testa_estado_da_maquina()
      {
            String str;
            try
            {

      //lcd.escreve(tira_xml_status(ServicosHttp.net(null,ip,
"status", false)));
                  str =
tira_xml_status(ServicosHttp.net("CentralTelemetria=" +
Menu.nome_central,Menu.ip, "status"));
                  System.out.println(str);
                  LCD_Mgr.pula_linha("1");
                  if(str.indexOf("não Inicializada") > 0)
                        LCD_Mgr.escreve_sem_pular_linha("Nao
Inicializada");
                  else

      LCD_Mgr.escreve_sem_pular_linha("Inicializada");
            }
            catch (IOException e)
            {
                  e.printStackTrace();
```

```
            }
            catch (GpioException e)
            {
                  e.printStackTrace();
            }
      }

      public static void detalhes_os(Vector nome_de_produto, Vector
quantidade_de_produto, int os_indice) {

            System.out.println("Nome do produto:" +
nome_de_produto.elementAt(os_indice));
            System.out.println("Quantidade do produto:" +
quantidade_de_produto.elementAt(os_indice));

            try{
            LCD_Mgr.LCD_clear_and_reset();

      LCD_Mgr.escreve_sem_pular_linha((String)nome_de_produto.elementA
t(os_indice));
            LCD_Mgr.pula_linha("1");

      LCD_Mgr.escreve_sem_pular_linha((String)quantidade_de_produto.el
ementAt(os_indice));
            }
            catch(Exception e)
            {
                  e.printStackTrace();
            }
      }

      public static void inicializar()
      {
            String str;
            try
            {
                  str =
tira_xml_inicializar(ServicosHttp.net("CentralTelemetria=" +
Menu.nome_central, Menu.ip, "inicializar"));
                  System.out.println(str);

                  LCD_Mgr.pula_linha("1");

                  if(str.indexOf("com Sucesso") > 0 )
                        LCD_Mgr.escreve_sem_pular_linha("Sucesso!");
                  else
                        LCD_Mgr.escreve_sem_pular_linha("Fail!");

                  Menu.inicializa = true;


            }
            catch (IOException e)
            {
                  e.printStackTrace();
            } catch (GpioException e) {

                  e.printStackTrace();
            }
      }
```

```
	private static String tira_xml_inicializar(String net) {//Ok!
		if(net.indexOf("<Sucesso>") > 0)
			return
net.substring(net.indexOf("<Sucesso><Mensagem>") + 19 ,
net.indexOf("</Mensagem><Da"));

		return net.substring(0, net.indexOf("<", 0)) + " - " +
		 net.substring(net.indexOf("<ID>") + 4,
net.indexOf("</ID>")) + " - " +
		 net.substring(net.indexOf("<Mensagem>") + 10,
net.indexOf("</Mensagem>"));
	}

	public static void executa_os_corrente(String nome_central,
String ip, String osid)
	{
		try {
			ServicosHttp.net("CentralTelemetria=" + nome_central
+ "&idOrdemServico="+osid,ip,"abastecida");
			} catch (IOException e) {

			e.printStackTrace();
		}
	}

	public static void completar()
	{
		try {
			ServicosHttp.net("CentralTelemetria=" +
Menu.nome_central,Menu.ip,"completa");
		} catch (IOException e) {
			e.printStackTrace();
		}
		System.out.println("Maquina com reservatorios
completos!!!");
	}

	public static void estado_sensores(boolean [] entrada)
	{
		try
		{
			ServicosGpio.le_tudo(entrada);
			for(int i = 0; i < 12; i++)
				if(entrada[i])
				{
					System.out.println("1");
					LCD_Mgr.escreve("1");
				}
				else
				{
					System.out.println("0");
					LCD_Mgr.escreve("0");
				}
		}
		catch (GpioException e)
		{
			e.printStackTrace();
		}
	}
```

```
	public static boolean pega_oss(String os, Vector
nome_de_produto, Vector quantidade_de_produto, Vector IDos, Vector
dataOs)
	{
		String data, id;

		try
		{
			os = ServicosHttp.net("CentralTelemetria=" +
Menu.nome_central,Menu.ip, "os");
		} catch (IOException e)
		{
			e.printStackTrace();
		}


		if(os.indexOf("<erro>") < 0)
		{
			os =
os.substring(os.indexOf("<OrdensDeServico>")+17,os.indexOf("</OrdensDe
Servico>"));

			nome_de_produto.removeAllElements();
			quantidade_de_produto.removeAllElements();
			IDos.removeAllElements();
			dataOs.removeAllElements();

			while(true)
			{
				if(os.length() <= 0)
					break;

				id =
os.substring(os.indexOf("<idOrdemServico>") + 16,
os.indexOf("</idOrdemServico>"));
				data = os.substring(os.indexOf("<Data>") + 6,
os.indexOf("</Data>"));
				data = data.replace('-', ' ');
				data = data.trim();

				pega_produtos(nome_de_produto,
quantidade_de_produto, IDos, dataOs,
os.substring(os.indexOf("<Produtos>") + 10,os.indexOf("</Produtos>")),
id, data);

				if(os.indexOf("</OS>") + 5 > 0)
					os = os.substring(os.indexOf("</OS>") +
5);
				else
					break;
			}


			System.out.println("\n\n-----------");
			System.out.println("Osid:" + IDos.elementAt(0));
			System.out.println("Data:" + dataOs.elementAt(0));
			System.out.println("-----------");

			for(int fim = 0; fim < nome_de_produto.size(); fim++)
			{
				if(fim > 0)
```

```
                                {
                                        if(IDos.elementAt(fim) !=
IDos.elementAt(fim-1))
                                        {
                                                System.out.println("\n\n-----
------");
                                                System.out.println("Osid:" +
IDos.elementAt(fim));
                                                System.out.println("Data:" +
dataOs.elementAt(fim));
                                                System.out.println("---------
--");
                                        }
                                }
                        System.out.println("Nome:" +
nome_de_produto.elementAt(fim));
                        System.out.println("Quantidade:" +
quantidade_de_produto.elementAt(fim));

                }
                return true;
        }
        else
        {
                System.out.println("Erro!");
                return false;
        }

    }

    private static void pega_produtos(Vector nomeDeProduto,
                Vector quantidadeDeProduto, Vector iDos, Vector
dataOs, String substring, String i, String data) {

        while(true)
        {
                if(substring.length() <= 0)
                        break;

    nomeDeProduto.addElement(substring.substring(substring.indexOf("
<Nome>") + 6,
                                substring.indexOf("</Nome>")));


    quantidadeDeProduto.addElement(substring.substring(substring.ind
exOf("<Quantidade>") + 12,
                                substring.indexOf("</Quantidade>")));

                        iDos.addElement(i);
                        dataOs.addElement(data);

                        if(substring.indexOf("</Produto>") + 10 > 0)
                                substring =
substring.substring(substring.indexOf("</Produto>") + 10);
                        else
                                break;

        }
    }

    public static int pega_os_anterior(int os_atual, Vector IDos) {
```

```
            //Retorna o indice da OS anterior.

            if(IDos.elementAt(os_atual) == IDos.elementAt(0)) //A id
atual é a primeira. Não tem como voltar mais.
                  return 0;

            for(int fim = IDos.size()-1; fim > 0; fim--)
            {
                  if((IDos.elementAt(fim - 1) !=
IDos.elementAt(os_atual)) && (IDos.elementAt(fim) ==
IDos.elementAt(os_atual)))
                        return fim - 1;
            }
            return -1; //Erro!
      }

      public static int pega_proxima_os(int os_atual, Vector IDos) {

            if(IDos.elementAt(os_atual) == IDos.elementAt(IDos.size()-
1)) //A id atual é a última. Não tem como ir mais.
                  return os_atual;

            //System.out.println(IDos.size());

            for(int fim = 1; fim < IDos.size()-1; fim++)
            {
                  //System.out.println("Fim + 1 = "+IDos.elementAt(fim
+ 1));
                  //System.out.println("OS atual =
"+IDos.elementAt(os_atual));
                  //System.out.println("fim = "+IDos.elementAt(fim));

                  if((IDos.elementAt(fim + 1) !=
IDos.elementAt(os_atual)) && (IDos.elementAt(fim) ==
IDos.elementAt(os_atual)))
                        {
                        //    System.out.println(" fim  = " + fim);
                              return fim + 1;
                        }
            }
            return -1; //Erro!
      }

      public static int pega_produto_anterior(Vector IDos, int
indice_atual) {

            if(indice_atual == 0)
                  return 0; // Não dá pra voltar mais...

            if(IDos.elementAt(indice_atual) ==
IDos.elementAt(indice_atual - 1)) //Se o elemento anterior for da
mesma OS
                  return --indice_atual;
            else
                  return indice_atual;//O próxiimo elemento nao é da
mesma OS. Estamos no ultimo elemento dessa OS
      }

      public static int pega_proximo_produto(Vector IDos, int
indice_atual) {
```

```
            if(indice_atual >= IDos.size()-1)
                  return indice_atual; // Não dá pra avançar mais...

            if(IDos.elementAt(indice_atual) ==
IDos.elementAt(indice_atual + 1)) //Se o próximo elemento for da mesma
OS
                  return ++indice_atual;
            else
                  return indice_atual;//O próxiimo elemento nao é da
mesma OS. Estamos no ultimo elemento dessa OS
      }

      public static void atualiza_lcd_com_o_produto_atual(produto el)
{

            String cart = "#", nome = "", id = "(";

            cart +=  el.cartucho;

            if(el.cartucho < 1000)
                  cart += " ";

            if(el.cartucho < 100)
                  cart += " ";

            if(el.cartucho < 10)
                  cart += " ";

            id += el.id_produto + ")";

            nome = el.nome;

            if (el.nome.length() > (10 - id.length()))
                  nome = nome.substring(0,(10 - id.length()));


            try {
                  LCD_Mgr.LCD_clear_and_reset();
                  LCD_Mgr.escreve(cart + " " +nome + id);
                  LCD_Mgr.pula_linha("1");
                  LCD_Mgr.escreve("EA: " + el.estoque_atual + " MAX: "
+ el.capacidade_max);

                  System.out.println("LCD:\n" + cart + " " +nome + id);
                  System.out.println("EA: " + el.estoque_atual + " MAX:
" + el.capacidade_max);
          } catch (GpioException e) {
                  e.printStackTrace();
            } catch (Exception e) {
                  e.printStackTrace();
            }

      }

}
```

4. Arquivo LCD_Mgr.java:

```
package g24;

import com.motorola.oem.hapi.GpioException;
import com.motorola.oem.hapi.GpioOutput;

public class LCD_Mgr {
	private static boolean[] saida = new boolean[8];
	private static int charcount = 0;
	public static GpioOutput g_write;
	public static GpioOutput g_clk_o;
	public static GpioOutput g_strobe;
	public static GpioOutput ge;
	public static GpioOutput grs;

	public static void inicia_lcd() throws Exception
	{
		//Coisas do Output
		g_clk_o = new GpioOutput(6);
		g_strobe = new GpioOutput(5);//Coloca os dados pra fora.
		g_write = new GpioOutput(7);
		grs = new GpioOutput(9);//RS
		ge = new GpioOutput(10);//E

		inicia();
	}

	private static void inicia() throws Exception
	{
		ge.write(false);
		grs.write(false);//comando

		LCD("00111100");

		LCD("00001111");

		LCD("00000001");

		LCD("00111100");

	}

	public static void LCD_clear_and_reset() throws Exception{
		grs.write(false);//comando

		LCD("00000001");
		LCD("10000000");
		charcount = 0;

		grs.write(true);
	}

	public static void pula_linha(String string) throws
GpioException {

		grs.write(false);//comando

		try {
			LCD("1"+string+"000000");
		} catch (Exception e) {
```

```
                e.printStackTrace();
            }

            ServicosGpio.pulse(ge);
      }

      public static void LCD(String s) throws Exception {
            if (s.length() < 8)
                  throw new Exception("Faltam parametros para a funcao
LCD!!");

            for (int i = 0; i < 8; i++) {
                  saida[7-i] = (s.charAt(i) == '1');
            }
            escreve_tudo();
            ServicosGpio.pulse(ge);
      }

      public static void escreve(String s) throws GpioException
      {

          byte[] b = s.getBytes();
          boolean[] bol;

          grs.write(true);//letra

          for ( int i = 0; i < s.length(); i++ )//para cada letra
          {
            if(charcount == 16)
            {
                  pula_linha("1");
                  grs.write(true);
                  if((s.charAt(i) == ' ') && (i + 1 < s.length()))
                        i++;
            }
            //System.out.println("   avg cc = " + charcount + " " + i);
            if(charcount == 32)
            {
                  pula_linha("0");
                  grs.write(true);
                  charcount = 0;
                  if((s.charAt(i) == ' ') && (i + 1 < s.length()))
                        i++;
            }

            charcount ++;

            bol = convert(b[i]);//letra em booleanos

            for(int j = 0; j < 8; j++)
                  saida[7-j] = bol[j]; //transfere pra saida

            escreve_tudo();
            ServicosGpio.pulse(ge);
      }
      }

      public static void escreve_sem_pular_linha(String s) throws
GpioException
      {
            byte[] b = s.getBytes();
```

```
        boolean[] bol;
        int fim = 16;

        if(fim > s.length()-1)
          fim = s.length()-1;

        grs.write(true);//letra

        for ( int i = 0; i <= fim; i++ )//para cada letra
        {
          bol = convert(b[i]);//letra em booleanos

          for(int j = 0; j < 8; j++)
                saida[7-j] = bol[j]; //transfere pra saida

          escreve_tudo();
          ServicosGpio.pulse(ge);
    }
    }

    private static void escreve_tudo() throws GpioException
    {
          int i = 0;

          for(i = 0; i < 8; i++)
          {
                g_write.write(saida[7-i]);
                ServicosGpio.pulse(g_clk_o);
          }
          g_strobe.write(true);
          g_strobe.write(false);
    }

    private static boolean[] convert(byte b) {
          boolean[] bits = new boolean[8];
      for (int i = 0; i < bits.length; i++) {
          bits[7-i] = ((b & (1 << i)) != 0);
      }
      return bits;
}

    public static void sensores(boolean[] entrada) {
          String s = "";
          for(int i = 0; i < 11; i++)
          {
                if(entrada[i])
                      s += "1";
                else
                      s += "0";

          }

          try {
                pula_linha("1");
                escreve_sem_pular_linha(s);
          } catch (GpioException e) {

                e.printStackTrace();
          }

    }
```

```
}
```

5. Arquivo produto.java:

```
package g24;

public class produto {

	public int id_produto, cartucho, estoque_atual, capacidade_max;
	public float preco;
	public byte MDBcode;
	public String nome;

	public produto(int id_produto, int cartucho, float preco, byte
MDBcode, String nome)
	{
		this.cartucho = cartucho;
		this.id_produto = id_produto;
		this.preco = preco;
		this.MDBcode = MDBcode;
		this.nome = nome;
	}
}
```

6. Arquivo ServicosGpio.java:

```
package g24;

import com.motorola.oem.hapi.GpioException;
import com.motorola.oem.hapi.GpioInput;
import com.motorola.oem.hapi.GpioInterruptConfig;
import com.motorola.oem.hapi.GpioOutput;

public class ServicosGpio {

	private static GpioInput g14 ;
	private static GpioInput g16;
	private static GpioInput g_read;
	public static GpioOutput g_clk_i;
	public static GpioOutput g_latch_input;
	public static GpioOutput wkupo;


	public static void inicia(GpioInterruptConfig
gpioInterruptConfig) throws GpioException
	{
		//Coisas do Input
		g_clk_i = new GpioOutput(2);//Clock
		g_latch_input = new GpioOutput(3);//OE
		g_read =  new GpioInput(4);//I

		wkupo = new GpioOutput(11);
		wkupo.write(false);

		//Interrupções
		g14 = new GpioInput(14); //Teclado
		g16 = new GpioInput(16); //Sensores
```

```
            g14.enableInterrupt(gpioInterruptConfig);
            g16.enableInterrupt(gpioInterruptConfig);
      }

      public static void pulse(GpioOutput g) throws GpioException
      {
            g.write(true);
            g.write(false);
      }

      public static void le_tudo(boolean entrada[]) throws
GpioException
      {
            int i = 0;

            g_latch_input.write(true);
            pulse(g_clk_i);
            g_latch_input.write(false);

            for(i = 0; i < 16; i++)
            {
                  entrada[i] = g_read.read();
                  pulse(g_clk_i);
            }
      }
}
```

7. Arquivo ServicosHttp.java:

```
package g24;

import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import javax.microedition.io.Connector;
import javax.microedition.io.HttpConnection;

import com.motorola.oem.network.Network;
import com.motorola.oem.websession.WebSession;
import com.motorola.oem.websession.WebSessionException;
import com.motorola.oem.websession.WebSessionManager;

public class ServicosHttp {

      static HttpConnection http;
      static byte[] b = new byte [32];
      //static HttpConnection httpConn = null;
      public static int cont_conexoes = 0;

      static void conectar()
      {
            System.out.println("GSM registration state: " +
gsmRegStateToString(Network.getGSMRegState()));

            WebSession wb = new WebSession();
        wb.setAccessType(WebSession.ACCESS_RW);
          wb.setGprsApn("zap.vivo.com.br");
          wb.setGprsPassword("vivo");
```

```
            wb.setGprsUserName("vivo");
            wb.setSessionName("Java Session");

            try {
                    WebSessionManager.updateWebSession(wb, 1);
              } catch (WebSessionException e1) {

                    e1.printStackTrace();
              } catch (NullPointerException e1) {

                    e1.printStackTrace();
              } catch (IllegalArgumentException e1) {

                    e1.printStackTrace();
              }
       }

        public static String gsmRegStateToString(int gsmRegState) {
             switch (gsmRegState) {
         case Network.GSM_STATE_NOT_REG_NOT_SEARCH:
             return "Not registered, not searching";

         case Network.GSM_STATE_NOT_REG_SEARCH:
             return "Not registered, searching";

         case Network.GSM_STATE_REG_DENIED:
             return "Registration denied";

         case Network.GSM_STATE_REG_HOME:
             return "Registered home";

         case Network.GSM_STATE_REG_ROAM:
             return "Registered roaming";

         case Network.GSM_STATE_UNKNOWN:
             return "Unknown";
     }

     throw new IllegalArgumentException("Illegal GSM registration
state");
}

       static String net(String parametro, String ip, String url)
throws IOException
       {
             String _url;
             InputStream is = null;
             OutputStream os = null;
             StringBuffer sb = new StringBuffer();

             HttpConnection httpConn = null;

             if(ip == null)
             {
                   _url = "http://tcc.gateon.com.br/?" + url;
             }
             else
             {
                   _url = "http://" + ip + "/tcc/remoto.php/?" + url;
             }
             try {
```

```
                    httpConn = (HttpConnection)Connector.open(_url);

                    httpConn.setRequestMethod(HttpConnection.POST);

                    httpConn.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded");

                  os = httpConn.openOutputStream();
                  System.out.println(parametro);

                  os.write(parametro.getBytes());

                //  System.out.println("+++++++++++++++++++ Tentando
Abrir ++++++++++++++++++");
                 // System.out.println("cont_conexoes = " +
++cont_conexoes);
                  is = httpConn.openInputStream();

                  int chr;

                  try
                  {
                    while ((chr = is.read()) != -1)
                          sb.append((char)chr);
                  }
                  catch (IOException sp)
                  {
                    System.out.println("Deu erro aqui!");
                    System.out.println("Di.available:" +
is.available());
                    System.out.println(sp);

                    //sp.printStackTrace();
                  }
                  // Web Server just returns the birthday in mm/dd/yy
format.
                  }
                        catch (Exception e)
                        {
                              System.out.println("-1!-");
                              e.printStackTrace();

                        }
                  try{

                    if(is!= null)
                    {
                   //   System.out.println("---------- Fechando IS ----
--------");
                        is.close();
                   //   System.out.println("---------- IS ok! ---------
---");
                    }
                      if(os != null)
                      {
                   //     System.out.println("---------- Fechando OS --
----------");
                        os.close();
                   //     System.out.println("---------- OS ok! -------
-----");
                      }
```

```
            }
                   catch(Exception e)
                  {
                        System.out.println("-2!-");
                        e.printStackTrace();
                  }

                  try{

                      if(httpConn != null)
                          {
                      //      System.out.println("---------- Fechando
HTTP ------------");
                              httpConn.close();
                           }
                     }
                         catch(IOException e)
                        {
                              System.out.println("-Não conseguiu
fechar!! Colocando NULL!-");
                              System.out.println(e);
                              httpConn = null;
                        }

             System.out.println(sb.toString());
               return sb.toString();
       }

       public static void produtos_desta_maquina() {//Coloca os
produtos que o servidor diz que tem nesta máquina no
                  //vetor produtos.

              String _url;
              OutputStream os = null;
              DataInputStream di = null;
              StringBuffer sb = new StringBuffer();
              HttpConnection httpConn;

             // byte[] b = new byte[1024];

              _url = "http://" + Menu.ip + "/tcc/remoto.php/?" +
"status";

              try {
                     httpConn = (HttpConnection)Connector.open(_url);

                     httpConn.setRequestMethod(HttpConnection.POST);

                     httpConn.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
                     httpConn.setRequestProperty("User-
Agent","Profile/MIDP-2.0 Confirguration/CLDC-1.1");
                  httpConn.setRequestProperty("Accept_Language","en-
US");

                  os = httpConn.openOutputStream();
                  os.write(("CentralTelemetria=" +
Menu.nome_central).getBytes());

                  di = httpConn.openDataInputStream();
```

```
                    String aux;
                    try
                    {
                      System.out.println("- Produtos na máquina - ");
                      while (di.available() > 1)
                      {
                            for(int i = 0; i < 900; i++)
                            {
                                  if(di.available() < 1)
                                      break;
                                        sb.append((char)di.read());

                            }//900  letras lidas

                            while(sb.toString().indexOf("</Cartucho>") >
0)
                                {
                                  aux = sb.toString();
                                  aux =
aux.substring(aux.indexOf("<Cartucho>") + 10,
aux.indexOf("</Cartucho>"));
                                  sb =
sb.delete(0,sb.toString().indexOf("</Cartucho>") + 10);
                                  PegaProduto(aux);
                            }
                      }
                    }
                    catch (Exception sp)
                    {
                      System.out.println("Deu erro aqui!");
                      System.out.println("Di.available:" +
di.available());
                      sp.printStackTrace();
                    }
                              di.close();

                    }
                          catch (Exception e)
                          {
                                System.out.println("-1!-");
                                e.printStackTrace();
                          }

                          try
                          {
                            if(di != null)
                                  di.close();
                        if(os != null)
                          os.close();
                          }
                          catch(Exception e)
                          {
                                System.out.println("-2!-");
                                e.printStackTrace();
                          }

                    System.out.println("Terminou Produtos Desta
Máquinaaaa!");
              }
```

```
            static void PegaProduto(String aux)
            {

                  int id_produto, cartucho;
                  float preco;
                  String nome;

                  produto p;

                  //System.out.println("Aux dentro de pegaproduto = " +
aux);

                  preco =
Float.parseFloat(aux.substring(aux.indexOf("<Preco>")+7,
aux.indexOf("</Preco>")));
                  id_produto =
Integer.parseInt(aux.substring(aux.indexOf("<idProduto>")+11,
aux.indexOf("</idProduto>")));
                  cartucho =
Integer.parseInt(aux.substring(aux.indexOf("<Numero>")+8,
aux.indexOf("</Numero>")));
                  nome = aux.substring(aux.indexOf("<Nome>")+6,
aux.indexOf("</Nome>"));

                  System.out.println("-> Nome: " + nome);
                  System.out.println("--> " +
aux.substring(aux.indexOf("<idProduto>"),aux.indexOf("<Detalhes>")));

                  byte b = 0;
                  p = new produto(id_produto, cartucho, preco,b, nome);
                  Menu.produtos_desta_maquina.addElement(p);


            }

      }
```

8. Arquivo Uart.java:

```
package g24;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import javax.microedition.io.CommConnection;
import javax.microedition.io.Connector;

public class Uart {

      static CommConnection commConectionUart1;
      static InputStream leitor;
      static OutputStream escritor;
      static byte[] Byte_lido = new byte[18];
      static byte[] b = new byte [1000];
      static byte bb;

      public static void inicia_uart1() throws IOException
      {
```

```
	commConectionUart1 =
(CommConnection)Connector.open("comm:COM1;baudrate=9600;autocts=off;au
torts=off", Connector.READ_WRITE, true);
			leitor = commConectionUart1.openInputStream();
			escritor = commConectionUart1.openOutputStream();
	}
	public static void megaprint (String s)
	{
			b = s.getBytes();
			try {
				escritor.write(b);
			} catch (IOException e1) {

				e1.printStackTrace();
			}
			System.out.println(s);
	}
	public static byte[] le_da_uart()
	{
			int i, cont = 0;
			System.out.println("Lendo!");
			try {
				if(leitor.available() > 0)
				{
					for(i = 0; i <Byte_lido.length; i++)
						Byte_lido[i] = 0;//Zerando os bytes lidos


					while(leitor.available() > 15)
					{//Lendo de 15 em 15 bytes
						leitor.read(Byte_lido, 0, 15);
						for(i = 0; i < 15; i++)
						{
							if(Byte_lido[i] != 0)
							{
								b [cont] =
Byte_lido[i];//Transformando em String
								cont++;
							}
							/**
							 * Caso os bytes do MDB não gerem
caracteres alfanuméricos, é
							 * só não converter para char.
Trabalhamos como INT, que não tem
							 * problema.
							 * */
						}
					}
					leitor.read(Byte_lido, 0, leitor.available());
					for(i = 0; i < Byte_lido.length; i++)
					{
						if(Byte_lido[i] != 0)
						{
							b [cont] =
Byte_lido[i];//Transformando em String
							cont++;
						}
					}

					return b;
```

```
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
            return b;
        }


        public static byte le_1_byte_da_uart()
        {
            int i;
            System.out.println("Lendo!");
            try {
                if(leitor.available() > 0)
                {
                    for(i = 0; i <Byte_lido.length; i++)
                        Byte_lido[i] = 0;//Zerando os bytes lidos

                    if(leitor.available() >= 1)
                        leitor.read(Byte_lido, 0, 1);

                    System.out.println("Byte lido:" + (char)bb);
                    return bb;

                }
            } catch (Exception e) {
                e.printStackTrace();
            }
            return bb;
        }
}
```

Anexo 2: Firmware do Conversor 9bits (MDB) -> 8bits

```
#include p16F876a.inc

   cblock 0x20
   cont
   aux
   tempo
   tempo2
         conta_2a_vez
   endc
   ser de 4 x 9600 = 38400Hz ~ 40Khz (no mínimo.)
   ;Usando um clock de 4MHz, sem pré-scaler, temos cada incremento do timer a 4000000 =
4MHz
   ;Assim, o timer tem que contar até 40000/384 = 104
   ;


   BANKSEL PR2
   movlw .104 ;Timer2 vai contar até este valor, para interromper em 9600hz, a partir do clock.


   movwf PR2

   ;setando interrupções
   movlw b'11010000'
   movwf INTCON ;Ta no mesmo banco do PR2!

         bcf OPTION_REG, INTEDG
   movlw b'00001111'
   movwf TRISB ;Ta no mesmo banco do PR2!
         clrf PORTC
   BANKSEL PORTB

         clrf PORTB
         clrf PORTC

         bsf PORTB,4
         bsf PORTB,5

pisca:
         BSF PORTB,6
         call aguarda2
         BCF PORTB,6
         call aguarda2
         decfsz cont
         goto pisca

         movlw 9
   movwf cont
```

```
   goto $;loop infinito


interrupcao:
   ;bsf PORTB,7

   btfss PIR1, TMR2IF;É timer?
   goto int_portB ;não era...

         bsf PORTC,0
   ;Timer Interrupt:
   bcf PIR1, TMR2IF

   clrf TMR2 ;zera timer 2
   ;loop x9:
   btfsc aux,0;pula se ainda nao passaram 9 ciclos
   goto ok9ciclos

transf:
   ;Transferir o valor:
   btfsc PORTB,0
   goto liga
   bcf PORTB,4 ;desliga
   goto p2

liga:
   bsf PORTB,4
p2: ;loop x9
   decfsz cont ;pula se chegar em 0 (passaram 9 ciclos)
   retfie

   bsf aux,0 ;passaram 9 ciclos
   retfie
ok9ciclos: ;Parte especial da rotina, que envia o stopbit e o MS para o G24

         ;bsf PORTB,7
   bcf T2CON, TMR2ON;desliga timer2
   movlw 9
   movwf cont
   clrf aux

   btfsc PORTB,0 ;Colocar M/S na porta
   goto ligaMS
   bcf PORTB,6

   goto p2MS

ligaMS:
   bsf PORTB,6
p2MS:
```

```
;       bcf PORTB,7

  BANKSEL PIE1
  bcf PIE1, TMR2IE;desabilita as interrupções de timer
  BANKSEL PORTB
  ;re-seta timer para próximo bit
  clrf TMR2 ;zera timer 2
  bsf PORTB,4;Forçando o Stop Bit
  bcf PORTB,5;Gera interrupção no G24
  ;call aguarda SUBSTITUIDO POR 3 AGUARDA 2
        call aguarda2
        call aguarda2
        call aguarda2
  bsf PORTB,5
        ;bcf PORTC,0
        bcf INTCON,INTF
  bsf INTCON,INTE;re-habilita as interrupções na portb

 ; bcf PORTB,7

        call aguarda2;

        retfie

int_portB:
        bcf INTCON,INTF;abaixa flag

        ;btfsc conta_2a_vez ,0 ;Se n estvier setado, pule (faça a interrupção)
        ;goto nao_faca_a_int
        ;bsf conta_2a_vez, 0

        ;interrupção:
        BANKSEL PIE1
        bsf PIE1, TMR2IE;liga interrupção timer 2
        BANKSEL PORTB
        bsf T2CON, TMR2ON;liga timer2
        bcf INTCON,INTE;desliga knterrupção da portb

        goto transf;transfere o SB

nao_faca_a_int:
        bcf conta_2a_vez, 0
        retfie

aguarda:
        movlw .200
        movwf tempo
dec:
        decfsz tempo, F
        goto dec
        return
```

```
aguarda2:
        movlw .30
        movwf tempo2
dec2:
        call aguarda
        decfsz tempo2, F
        goto dec2
        return
        end
```

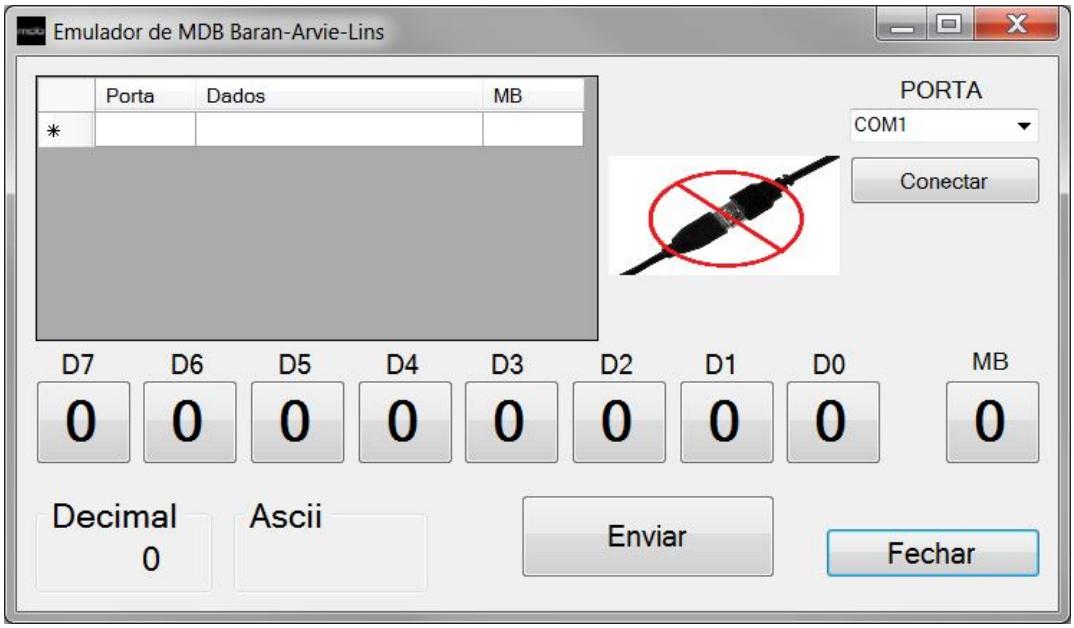
Anexo 3: Emulador MDB:

1. *Arquivo Serial.cs:*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO.Ports;
using System.Collections;

namespace EmuladorMDB
{
    public class Serial
    {
        SerialPort _serialPort;
        public bool available;

        public Serial()
        {
            _serialPort = new SerialPort();
            _serialPort.ReadTimeout = 500;
            _serialPort.WriteTimeout = 500;
            _serialPort.BaudRate = 9600;
        }

        public string GetPortName()
        {
            return _serialPort.PortName;
        }

        public bool StartConn()
        {
            try
            {
                _serialPort.Open();
            }
            catch(Exception nc)
            {
                System.Windows.Forms.MessageBox.Show("Erro na abertura da porta:
" + _serialPort.PortName + " codes:" + nc);
            }
            if (_serialPort.IsOpen)
            {
                available = true;
                return true;
            }
            else
            {
                available = false;
                return false;
            }
        }

        public bool CloseConn()
        {
            try
            {
                _serialPort.Close();
            }
            catch (Exception nc)
            {
```

```
                System.Windows.Forms.MessageBox.Show("Erro no fechamento da porta
: " + _serialPort.PortName + " codes:" + nc);
            }
            if (_serialPort.IsOpen)
            {
                available = true;
                return false;
            }
            else
            {
                available = false;
                return true;
            }
        }

        public ArrayList GetPortAvailable()
        {
            ArrayList Portas = new ArrayList();
            foreach (string s in SerialPort.GetPortNames())
            {
                Portas.Add(s);
            }

            return Portas;

        }

        public string SetPortName(string defaultPortName)
        {
            _serialPort.PortName = defaultPortName;
            return _serialPort.PortName;
        }

        public bool SetPortParity(int Paridade)
        {
            if (Paridade == 0)
                _serialPort.Parity = Parity.Space;
            else
                if (Paridade == 1)
                    _serialPort.Parity = Parity.Mark;
                else
                    return false;

            return true;
        }

        public bool EnviaByte(byte[] teste)
        {
            try
            {
                _serialPort.Write(teste, 0, 1);
            }
            catch (Exception envio)
            {
                System.Windows.Forms.MessageBox.Show("Erro no envio " + envio);
            }
            return true;
        }

    }
}
```

2. Arquivo Form1.Designer.cs:



Anexo 4: Layouts e Esquemáticos das placas:

1. *Gerenciadora de Entradas:*

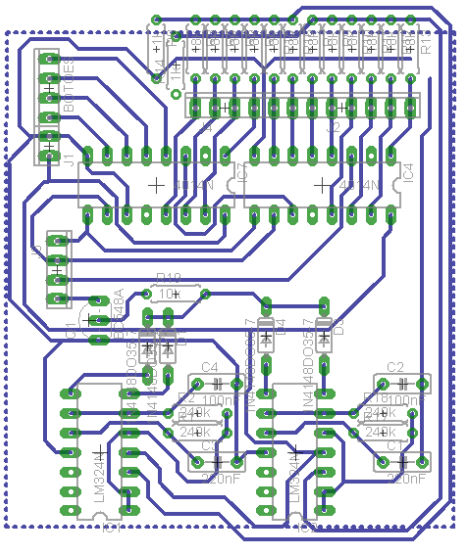


Figura 38 - Gerenciadora de Placas – Layout

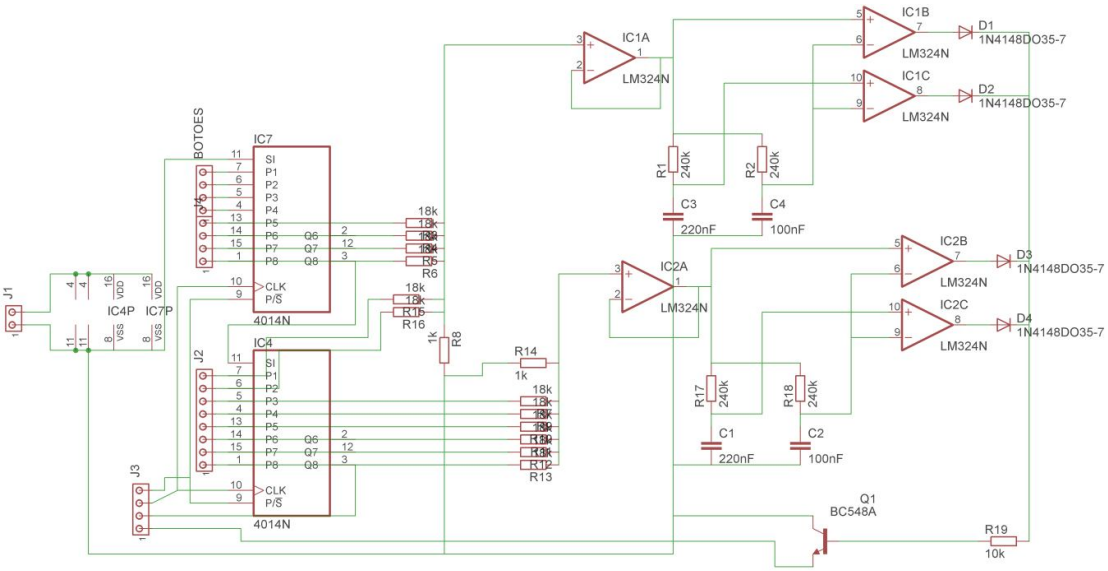


Figura 39 - Gerenciadora de Entradas – Esquema

2. Gerenciadora de Saídas:

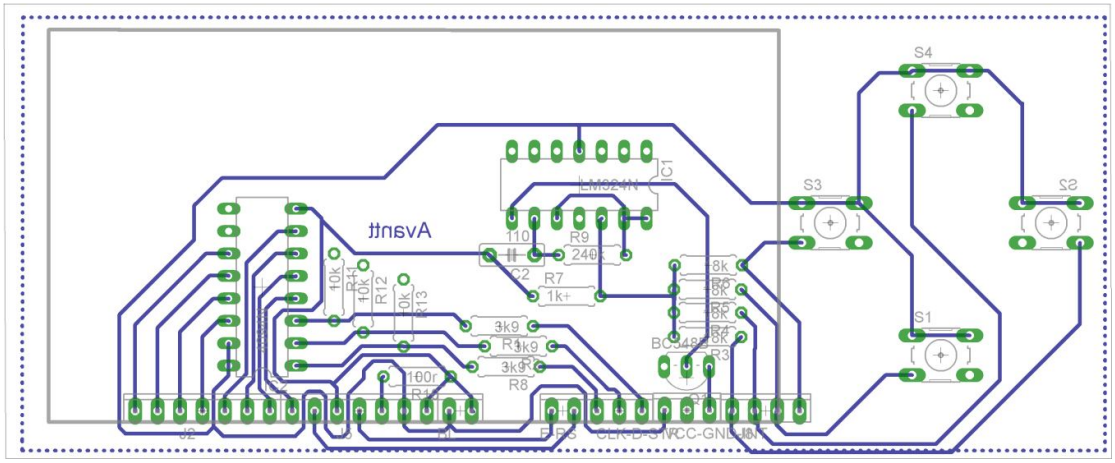


Figura 40 - Gerenciador de Saídas - Layout

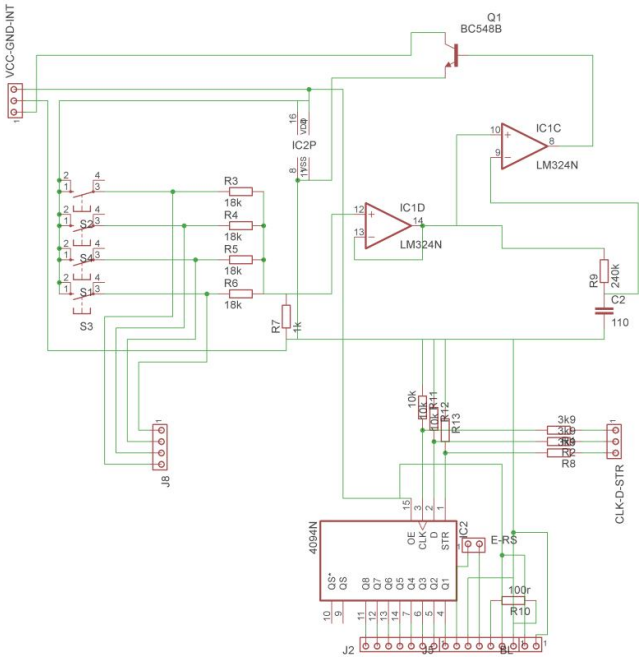


Figura 41 - Gerenciadora de Saídas – Esquemático

3. Principal:

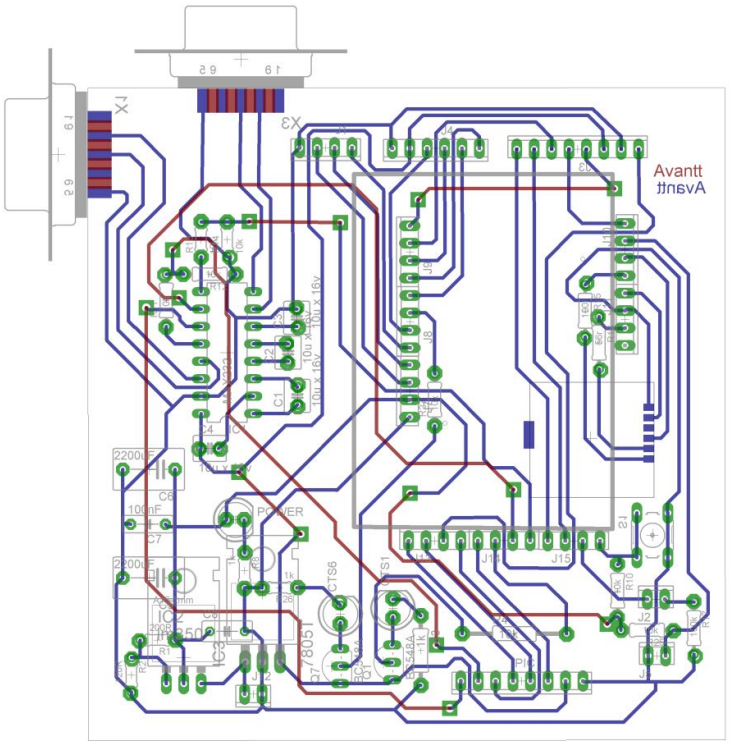


Figura 42 - Placa Principal - Layout

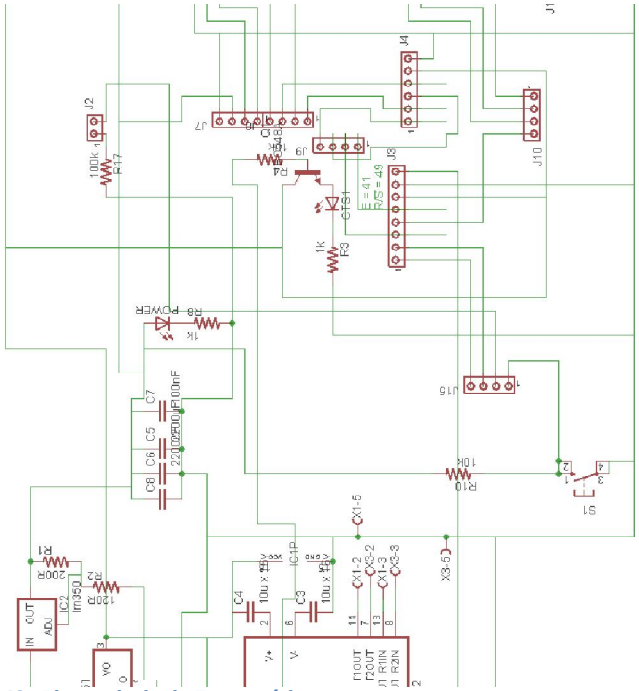


Figura 43 - Placa principal - Esquemático

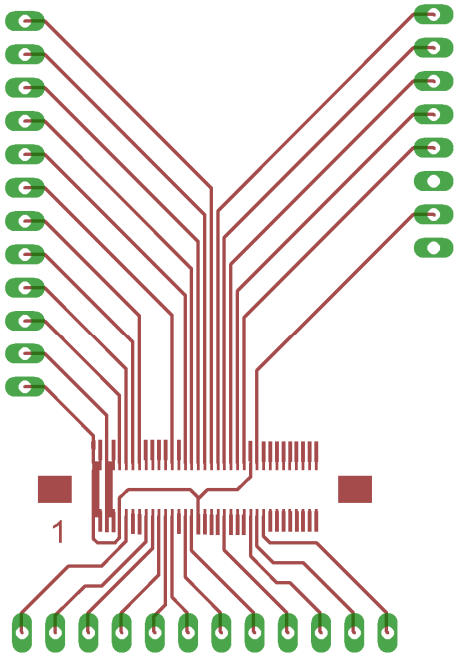


Figura 44 - Suporte do G24 - Layout

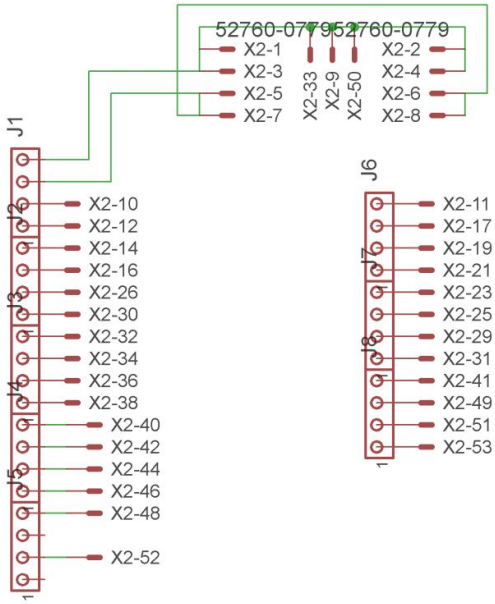


Figura 45 - Suporte do G24 – Esquemático

4. Conversor MDB -> 8bits

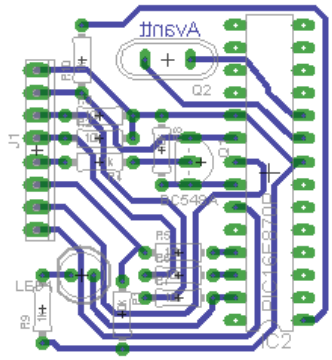


Figura 46 - Conversor MDB -> 8bits

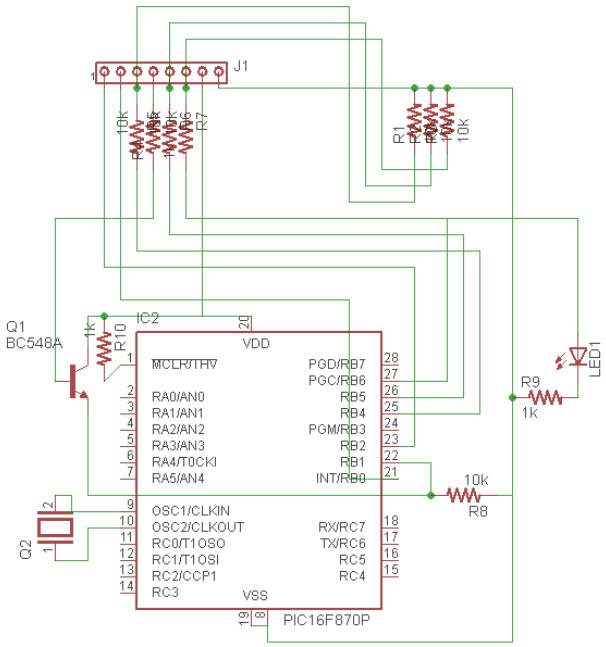


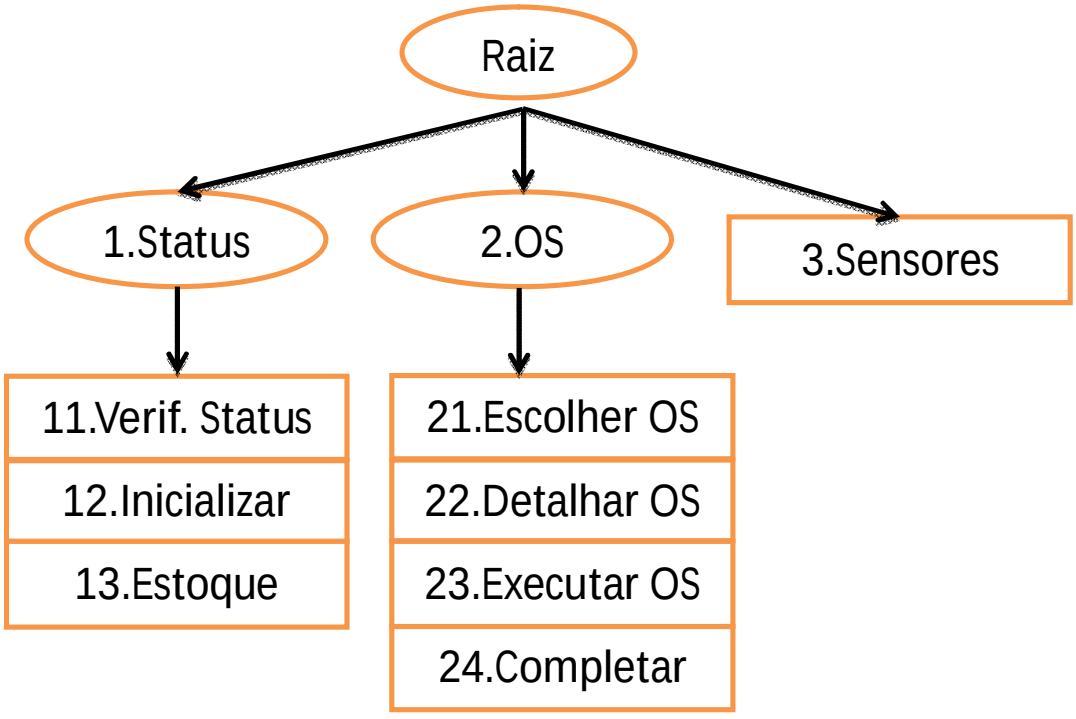
Figura 47 - Conversor MDB -> 8bits - Esquemático

Anexo 5: Manual de Instruções:

Manual de instruções da Central de Telemetria para Vending Machines

A central de telemetria tem um sistema de menus simples para se interagir. Ele é formado por quatro botões, sendo eles: Para cima, Para baixo, Voltar e Entrar. Para navegar nos menus, utilize as teclas Para Cima e Para Baixo. Para aceder a uma função, ou subir um nível no menu, utilize a tecla Entrar. Para retornar ao nível anterior, utilize Voltar.

O menu está estruturado conforme a árvore à seguir:



Os comandos possíveis da Central estão descritos abaixo:

1. *Inicialização da central:*

Quando a central é iniciada, ela recebe do servidor todos os itens que compõe o estoque da máquina. Para fazer a inicialização, alguns registros devem ser feitos na central, para que ela saiba qual produto está localizado em cada cartucho da vending machine. Este registro é feito de forma simples e automática na central, e necessita apenas de uma pequena intervenção do responsável técnico. Para executar este procedimento, siga os passos adiante:

a. No LCD, pressione o botão VOLTAR sucessivas vezes, até que apareça escrita a palavra “-RAIZ-”.
b. Pressione o botão ENTRAR. A central fará uma conexão com o servidor, e carregará as informações necessárias.
   Deverá aparecer no LCD a palavra “MENU”. Na linha de baixo, aparecerá a primeira opção: “1-Status”.
c. Aperte novamente o botão ENTRAR. O LCD deverá mostrar 11-VERIF. STATUS.
d. Usando as setas de direção, navegue até a opção 12 – INICIALIZAR.
e. Aperte novamente o botão ENTRAR.
f. O LCD mostra então a palavra “-Inicializa-”. Aguarde um momento enquanto a central se comunica com o servidor para iniciar os procedimentos de inicialização remota.
g. O LCD então deve mostrar a palavra “SUCESSO”, na segunda linha, indicando que a central foi corretamente inicializada. Caso apareça a palavra “-FAIL-”, reinicie o processo, a partir do início.
h. Após a terceira tentativa fracassada de inicialização, verifique as conexões elétricas da central, e/ou contate o suporte técnico.
i. Pressione novamente o botão “MENU”.
j. O LCD deverá exibir a mensagem “PRESSIONE BOTÃO”. Na linha inferior, deve aparecer um número, correspondente ao código de um dos produtos que devem compor o estoque da vending machine. Caso nenhum número apareça, ou a mensagem “PRESSIONE BOTÃO” não esteja visível, provavelmente não existem produtos cadastrados no servidor para esta vending machine. Contate o Suporte Técnico.
k. Pressione então o botão da vending machine referente ao produto cujo ID aparece na tela.
l. Repita os passos j. e k. até que todos os produtos estejam cadastrados na máquina.
m. Pronto. Ao fim do processo, o sistema deve voltar ao ENTRAR inicial. O LCD deve exibir a mensagem “1-STATUS”.

2. Verificação do Status

O Status da máquina indica se ela está inicializada ou não, junto ao servidor. Para verificar o status, siga os próximos passos:

a. No LCD, pressione o botão VOLTAR sucessivas vezes, até que apareça escrita a palavra “-RAIZ-”.
b. Pressione o botão ENTRAR. A central fará uma conexão com o servidor, e carregará as informações necessárias.
   Deverá aparecer no LCD a palavra “MENU”. Na linha de baixo, aparecerá a primeira opção: “1-Status”.
c. Aperte novamente o botão ENTRAR. O LCD deverá mostrar 11-VERIF. STATUS.
d. Aperte mais uma vez o botão ENTRAR. Aguarde um momento enquanto a central se conecta ao servidor, para carregar as informações necessárias.

e. O LCD então exibirá a mensagem “Inicializada”, confirmando a inicialização da máquina. Se a mensagem “Não inicializada” for exibida, então a central ainda não foi inicializada. Para inicializa-la, siga os passos indicados na seção 1.

3. Verificação de estoque

Devido ao grande número de dados exibidos, o estoque pode ser verificado com a ajuda de um computador com uma porta serial habilitada. As configurações da porta serial são as seguintes:

- Baud-rate 19200bps
- Controle de fluxo: off
- Número de Bits: 8
- Paridade: Nenhum

a. Conecte a porta serial do computador na porta número 2 da placa, através de um cabo serial sem “*cross-over*”.
b. Abra um cliente terminal no computador (Exemplo: Hercules ou Hyperterminal)
c. No LCD, pressione o botão VOLTAR sucessivas vezes, até que apareça escrita a palavra “-RAIZ-”.
d. Pressione o botão ENTRAR. A central fará uma conexão com o servidor, e carregará as informações necessárias.
Deverá aparecer no LCD a palavra “MENU”. Na linha de baixo, aparecerá a primeira opção: “1-Status”.
e. Pressione o botão ENTRAR mais uma vez. O LCD deve exibir “11.Verif. Status”.
f. Usando as setas de direção, navegue até a opção “13.Estoque”.
g. Pressione o botão ENTRAR. O LCD deve exibir “-Estoque-”. Agora a central vai transferir os dados para o computador.

Para verificar os dados no LCD da própria máquina, pressione as setas de direção para cima e para baixo. Os dados deverão surgir no LCD da seguinte forma:

#2 Azera p(4)
EA: 0 MAX: 5

Eles estão organizados da seguinte forma:

- O primeiro número representa o cartucho da máquina. No caso, 2.
- A palavra que aparece no final é o nome do produto, aqui, Azera p. Este nome pode aparecer cortado, se for muito grande e não "couber" no LCD. Isto é normal. Em nosso caso, o nome completo do produto é “Azera preto”.
- O segundo número, representa o ID do produto, em nosso exemplo, 4.
- Na linha de baixo, o número que vem após EA: é o Estoque Atual da máquina. No caso, temos 0 unidades.

- O número após MAX: indica a capacidade máxima da máquina. Aqui, temos 5 unidades, no máximo.

4. Ordens de Serviço

As Ordens de serviço são conjuntos de dados enviados pelo servidor para a Central de Telemetria, contendo os dados a respeito dos produtos que deverão ser reabastecidos na Vending Machine.

Quando o responsável pela execução da Ordem de serviço estiver junto à máquina, este pode verificar então os dados, através do sistema, para conferir com aqueles presentes na Ordem de Serviço a se executada. Para tanto, os seguintes passos devem ser executados:

a. No LCD, pressione o botão VOLTAR sucessivas vezes, até que apareça escrita a palavra "-RAIZ-".
b. Pressione o botão ENTRAR. A central fará uma conexão com o servidor, e carregará as informações necessárias.
   Deverá aparecer no LCD a palavra "MENU". Na linha de baixo, aparecerá a primeira opção: "1-Status".
c. Usando as setas de direção, navegue até a opção 2 = OS.
d. Aperte novamente o botão ENTRAR.
e. Aguarde um momento enquanto o sistema se conecta ao servidor, carregando as informações necessárias.
f. Caso apareça no LCD "NÃO TEM OS", significa que não existem ordens de serviço cadastradas no servidor. Se alguma ordem tenha sido emitida, mas não esteja aparecendo no sistema, contate o Suporte Técnico.
g. Quando as ordens de serviço disponíveis tiverem sido carregadas, o LCD exibirá "21.Escolher OS". Pressione o botão ENTRAR, para ver uma lista das Ordens de Serviço disponíveis para a máquina. As ordens de serviço são escolhidas pelo número de identificação (ID).
h. Após ser pressionado o botão ENTRAR, o LCD exibirá a palavra "Escolha", e na linha abaixo aparecerá um dos identificadores da Ordem de Serviço.
i. Navegue, usando as setas de direção, pelas OSs disponíveis, até que o identificador exibido seja idêntico ao presente no documento da Ordem de Serviço. Quando encontrar, pressione o botão VOLTAR. Obs.: Normalmente existe apenas um identificador, pois geralmente apenas uma ordem de serviço é gerada para a máquina de cada vez.
j. O ENTRAR "21.escolher OS" deve ter surgido novamente. Pressione as setas de direção, até que a opção "22.Detalhar OS" seja exibida. Pressione novamente o botão ENTRAR.
k. Agora o LCD deve exibir "-Detalhar OS-". Use as setas de navegação para exibir os produtos que compõe a Ordem de Serviço escolhida.
   Os produtos terão seu nome exibido na linha de cima, e a quantidade na linha de baixo.

l. Após verificar se todos os produtos da Ordem de Serviço conferem com os que estão no sistema, pode proceder com o reabastecimento. Depois, pressione o botão VOLTAR e navegue usando as setas de direção até a opção “23.Executar OS”.
m. Pressione o botão ENTRAR para finalizar a operação.

5. Sensores

A situação de cada sensor da vending machine, pode ser exibido no LCD. Para visualizar a situação dos sensores siga os passos a seguir:

a. No LCD, pressione o botão VOLTAR sucessivas vezes, até que apareça escrita a palavra “-RAIZ-”.
b. Pressione o botão ENTRAR. A central fará uma conexão com o servidor, e carregará as informações necessárias.
   Deverá aparecer no LCD a palavra “MENU”. Na linha de baixo, aparecerá a primeira opção: “1-Status”.
c. Usando as setas de direção, navegue até a opção 3 – SENSORES.
d. Pressione o botão ENTRAR. A central irá escrever no LCD a palavra “!ENTRADA!” na primeira linha, seguido da situação de cada sensor na segunda linha.
e. Os valores exibidos estão em forma de 1 ou 0. Quando exibido o valor 1, o sensor encontra-se ligado, e quando exibido o valor 0 o sensor encontra-se desligado. A ordem de exibição dos sensores no LCD vai da esquerda para direita em ordem crescente, ou seja, o primeiro sensor exibido é o sensor um e assim por diante.

6. Completando reservatórios

Para facilitar o processo de utilização da central, ela possui uma opção que completa o reservatório de todos os cartuchos junto ao servidor. Para efetuar tal processo os seguintes passos devem ser seguidos:

a. No LCD, pressione o botão VOLTAR sucessivas vezes, até que apareça escrita a palavra “-RAIZ-”.
b. Pressione o botão ENTRAR. A central fará uma conexão com o servidor, e carregará as informações necessárias.
   Deverá aparecer no LCD a palavra “MENU”. Na linha de baixo, aparecerá a primeira opção: “1-Status”.
c. Usando as setas de direção, navegue até a opção 2 – OS.
d. Aguarde um momento enquanto o sistema se conecta ao servidor, carregando as informações necessárias.

e. Caso apareça no LCD “NÃO TEM OS”, significa que não existem ordens de serviço cadastradas no servidor. Se alguma ordem tenha sido emitida, mas não esteja aparecendo no sistema, contate o Suporte Técnico.
f. Quando as ordens de serviço disponíveis tiverem sido carregadas, o LCD exibirá “21.Escolher OS”. Usando as setas de direção, navegue até a opção “24.Completar”. Pressione o botão ENTRAR, para iniciar a transação com o servidor.
g. Em caso de sucesso o LCD irá exibir a palavra “COMPLETA”. Em caso de insucesso tentar novamente, após alguns instantes. Caso o erro persista, entrar em contato com o Suporte Técnico para verificação do problema.

Manual de Integração

Sistema de Telemetria para Vending Machine

Este documento visa apresentar funcionalidades disponíveis no sistema para coleta de dados por aplicativos terceiros.

As informações poderão ser posteriormente processadas e organizadas a gosto do usuário e ou desenvolvedor de sistemas.

Conteúdo

Sistema de Telemetria para Vending Machine....115
Abertura de Conexão ao Servidor....115
Funções Remotas....116
Vending machine / Produto / Fornecedor....116
Vendingmachine....116
Produto....117
Fornecedor....117
Atualizar....118
Usuario....118
VendingMachine....119
Produto....119
Status....119
OS....120
Vendas....120
Alertas....123

Abertura de Conexão ao Servidor

O sistema de telemetria está configurado por trás de um Proxy que redireciona as conexões ao servidor de dados.

Para localizar o endereço do servidor de dados é necessário realizar uma conexão HTTP com o endereço HTTP://tcc.gateon.com.br/?ip. O retorno será uma string com formatação XML conforme o exemplo abaixo.

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<Conteudo>
```

```
        <Sucesso>
            <IP>187.59.66.145</IP>
        </Sucesso>
</Conteudo>
```

Uma vez obtido o endereço IP do servidor, futuras conexões deverão ser feitas ao endereço HTTP://**xxx.xxx.xxx.xxx**/remoto.php?FUNCAO, onde *"xxx.xxx.xxx.xxx"* representa o endereço IP da primeira chamada, e *"FUNCAO"* é o nome da função de interesse a ser executada.

Todas as funções exigem que sejam enviados parâmetros via POST para que as operações sejam corretamente executadas. A biblioteca ***cURL*** (http://curl.haxx.se/) poderá ser utilizada como auxílio na comunicação em aplicações desenvolvidas em PHP ou C, por exemplo. Bibliotecas similares podem estar disponíveis para outras linguagens de programação.

Funções Remotas

Abaixo estão listadas as funções remotas disponíveis para utilização. Elas deverão ser utilizadas no lugar de *"FUNCAO"* durante a requisição. Exemplo: HTTP://xxx.xxx.xxx.xxx/?STATUS.

O retorno é feito em forma de uma string com formatação XML. Caso haja um erro no processamento da operação, o campo Alerta contará com informações sobre o erro. Estas informações deverão ser utilizadas para o auxílio durante o processo de integração.

Exemplo:

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<Conteudo>
        <Alertas>
          <erro>
             <ID>-1</ID>
             <Mensagem>Operação Inválida</Mensagem>
          </erro>
        </Alertas>
</Conteudo>
```

Vending Machine / Produto / Fornecedor

Retorna os registros do usuário para cada uma das funções.

- POST
    - **Usuario** = Nome de usuário para acesso ao painel administrativo
    - **Senha** = Senha do usuário

Exemplos de resposta:

Vendingmachine

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<Conteudo>
  <Alertas />
```

```
  <Sucesso>
    <Mensagem>Registros do Usuário</Mensagem>
    <Vendingmachines>
      <Vendingmachine> -- 1..n estruturas
        <idVendingMachine>1</idVendingMachine>
        <CentralTelemetria>assdfee2288</CentralTelemetria>
        <Nome>Vending Machine n1</Nome>
        <Cartuchos>10</Cartuchos>
        <Logadouro>Rua Bruno Filgueira</Logadouro>
        <Cidade>Curitiba</Cidade>
        <CEP>80240220</CEP>
        <Estado>PR</Estado>
      </Vendingmachine>
    </Vendingmachines>
  </Sucesso>
</Conteudo>
```

Produto

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<Conteudo>
  <Alertas />
  <Sucesso>
    <Mensagem>Registros do Usuário</Mensagem>
    <Produtos>
      <Produto> -- 1..n estruturas
        <idProduto>1</idProduto>
        <Nome>produtox Teste 2</Nome>
        <PrecoCompra>10.00</PrecoCompra>
        <PrecoVenda>14.00</PrecoVenda>
        <Quantidade>50</Quantidade>
        <Descricao>Uma descrição de um produto interessante!</Descricao>
        <idFornecedor>0</idFornecedor>
      </Produto>
    </Produtos>
  </Sucesso>
</Conteudo>
```

Fornecedor

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<Conteudo>
  <Alertas />
  <Sucesso>
    <Mensagem>Registros do Usuário</Mensagem>
    <Fornecedors>
      <Fornecedor> -- 1..n estruturas
        <idFornecedor>1</idFornecedor>
```

```
        <Empresa>Garauco Exportação e Importação Ltda</Empresa>
        <CNPJ>07700061000100</CNPJ>
        <InscricaoEstadual>9035651281</InscricaoEstadual>
        <Logadouro>Rua Baltazar Carrasco dos Reis 2790</Logadouro>
        <Cidade>Curitiba</Cidade>
        <Estado>PR</Estado>
        <CEP>80250130</CEP>
        <ContatoNome>Glauco Lins</ContatoNome>
        <ContatoTelefone>(41)3941-4526</ContatoTelefone>
        <ContatoEmail>Glauco@Garauco.com.br</ContatoEmail>
      </Fornecedor>
    </Fornecedors>
  </Sucesso>
</Conteudo>
```

Atualizar

Permite atualizar no servidor informações do usuário.

- POST
  - **Usuario** = Nome de usuário para acesso ao painel administrativo
  - **Senha** = Senha do usuário
  - **Operacao** = Operação de atualização a ser realizada

As operações disponíveis e suas respectivas variáveis de POST estão definidas abaixo.

Campos entre colchetes são de preenchimento opcional. Todos os campos obrigatórios devem ser preenchidos, mesmo que os dados não sofram alteração.

Quando um campo é descrito como "Array", os dados são correspondentes entre campos do mesmo tipo. Exemplo: Produto[1], Capacidade[1] e Minimo[1] correspondem a informações do mesmo item.

As operações não geram retorno em formato XML no caso de sucesso e deverão ser verificadas através das funções de consulta. Quando houver retorno em formato XML é decorrente de algum erro na passagem dos parâmetros.

Usuario

```
  o  Usuario    => Novo nome de usuário para acesso ao sistema
  o  [Senha1]   => Nova senha
  o  [Senha2]   => Nova senha
  o  E-Mail     => Email (para envio de notificações)
  o  Empresa    => Empresa
  o  CNPJ       => CNPJ
  o  IE         => Inscrição Estadual
  o  Endereco   => Endereço
  o  Cidade     => Cidade
  o  Estado     => Unidade Federativa
  o  CEP        => CEP
  o  Pessoa     => Pessoa de Contato
  o  E-Mail2    => Email da Pessoa de Contato
  o  DDD        => DDD
```

- Telefone => Telefone

VendingMachine

- idVendinMachine => ID da vending machine a ser editada
- Nome => Nome da vending machine
- Logadouro => Endereço da vending machine
- Cidade => Cidade
- Estado => Unidade Federativa
- CEP => CEP
- Produto => Array – ID do Produto
- Capacidade => Array – Capacidade de itens no cartucho
- Minimo => Array – Estoque mínimo para emissão de OS

Produto

- idProduto => ID do produto a ser editado
- Nome => Nome do Produto
- PrecoCompra=> Preço de compra do produto
- PrecoVenda => Preço de venda do produto
- Quantidade => Quantidade disponível no armazém
- Descricao => Descrição do produto
- idFornecedor => ID do fornecedor

Status

Retorna o estado da Central de Telemetria/Vending Machine informada.

- POST
  - **CentralTelemetria** = Código da Central de Telemetria de interesse

Exemplo de resposta:

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<Conteudo>
      <Alertas />
      <Sucesso>
        <Mensagem>Situação do Estoque Registrado no
      Servidor</Mensagem>
        <Data>2010-11-19 05:20:28</Data>
        <nCartuchos>10</nCartuchos>
        <Cartuchos>
            <Cartucho>  -- 1..nCartuchos estruturas
              <Numero>1</Numero>
              <idProduto>1</idProduto>
              <Estoque>
                 <Capacidade>50</Capacidade>
                 <Atual>0</Atual>
                 <Minimo>20</Minimo>
              </Estoque>
              <Detalhes>
                 <Nome>produtox Teste 2</Nome>
                 <Preco>14.00</Preco>
                 <Descricao>Uma descrição de um produto</Descricao>
```

```
            </Detalhes>
          </Cartucho>
        </Cartuchos>
      </Sucesso>
</Conteudo>
```

OS

Retorna as ordens de serviço registradas para a Central de Telemetria/Vending Machine informada.

- POST
  - **CentralTelemetria** = Código da Central de Telemetria de interesse

Exemplo de resposta:

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<Conteudo>
  <Alertas />
  <Sucesso>
      <Mensagem>Resumo das Ordens de Servico</Mensagem>
      <OrdensDeServico>
        <OS>      -- 1..n estruturas
          <Data>2010-11-18 08:17:47</Data>
          <idOrdemServico>87</idOrdemServico>
          <Produtos>
            <Produto>    -- 1..n estruturas
              <idProduto>1</idProduto>
              <Cartucho>1</Cartucho>
              <Nome>produtox Teste 2</Nome>
              <Quantidade>50</Quantidade>
            </Produto>
            <Produto>
              <idProduto>1</idProduto>
              <Cartucho>3</Cartucho>
              <Nome>produtox Teste 2</Nome>
              <Quantidade>15</Quantidade>
            </Produto>
          </Produtos>
        </OS>
      </OrdensDeServico>
  </Sucesso>
</Conteudo>
```

Vendas

Retorna o histórico de vendas da Central de Telemetria/Vending Machine, dentro do período informado.

- POST
    - **CentralTelemetria** = Código da Central de Telemetria de interesse
    - [Opcional] **DataInicial** = AAAA-MM-DD
    - [Opcional] **DataFinal** = AAAA-MM-DD

Exemplo de resposta:

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<Conteudo>
  <Alertas />
  <Sucesso>
      <Mensagem>Resumo das Vendas</Mensagem>
      <DataInicial />
      <DataFinal />
      <Vendas>
        <Venda> -- 1..n estruturas
          <idProduto>2</idProduto>
          <Nome>Prod1</Nome>
          <Preco>6.64</Preco>
          <DataHora>2010-06-05 09:48:04</DataHora>
        </Venda>
        <Venda>
          <idProduto>2</idProduto>
          <Nome>Prod1</Nome>
          <Preco>90.38</Preco>
          <DataHora>2010-10-27 09:48:04</DataHora>
        </Venda>
      </Vendas>
  </Sucesso>
</Conteudo>
```

Alertas

Retorna o histórico de vendas da Central de Telemetria/Vending Machine, dentro do período informado.

- POST
    - **CentralTelemetria** = Código da Central de Telemetria de interesse
    - [Opcional] **DataInicial** = AAAA-MM-DD
    - [Opcional] **DataFinal** = AAAA-MM-DD

Exemplo de resposta:

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<Conteudo>
  <Alertas />
  <Sucesso>
     <Mensagem>Histórico de Alertas</Mensagem>
     <DataInicial />
```

```
      <DataFinal />
      <Alertas>
        <Alerta> -- 1..n estruturas
          <Mensagem>Máquina Inicializada</Mensagem>
          <DataHora>2010-10-03 03:26:01</DataHora>
        </Alerta>
      </Alertas>
    </Sucesso>
</Conteudo>
```